PB90-213893

DEGADIS (DEnse GAs DISpersion) Model Version 2.1. User's Guide

PEI Associates, Inc., Cincinnati, OH

Prepared for:

Environmental Protection Agency, Research Triangle Park, NC

Nov 89

U.S. Department of Commerce
National Technical Information Service
NTIS

PB90-213893

REPRODUCED BY
U.S. DEPARTMENT OF COMMERCE
NATIONAL TECHNICAL INFORMATION SERVICE
SPRINGFIELD, VA. 22161

(Please read Instructions on the reverse before completing)

1. REPORT NO. EPA/SW/DK-90/034a	2. PB90-213893	3. RECIPIENT'S ACCESSION NO.
4. TITLE AND SUBTITLE DEGADIS (DEnse GAs DISpersion) - Version 2.1		5. REPORT DATE June 1989
		6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S) T. O. Spicer and J. A. Havens		8. PERFORMING ORGANIZATION REPORT NO.
9. PERFORMING ORGANIZATION NAME AND ADDRESS		10. PROGRAM ELEMENT NO.
		11. CONTRACT/GRANT NO. EPA Contract #68-02-4351
12. SPONSORING AGENCY NAME AND ADDRESS U.S. Environmental Protection Agency Office of Air Quality Planning and Standards Source Receptor Analysis Branch Research Triangle Park NC 27711		13. TYPE OF REPORT AND PERIOD COVERED
		14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

EPA Project Officer: Dave Guinnup . For diskette see: PB90-502253

16. ABSTRACT

An improved Jet-Plume model has been interfaced with DEGADIS to provide for prediction of the trajectory and dilution of elevated dense gas jets to ground contact. DEGADIS predicts the ensuing ground-level plume dispersion. The Jet-Plume model provides for:

--automatic adjustment of integration step-size (using the Runge-Kutta-Gill method as in DEGADIS);

--elliptical plume shape (cross-section), with air entrainment specified consistent with the Pasquill-Gifford plume dispersion coefficient representation of atmospheric turbulent entrainment;

--user specification of averaging time;

--ground reflection when the plume (lower) boundary reaches ground level; and

--application to scenarios where the plume remains aloft.

17. KEY WORDS AND DOCUMENT ANALYSIS

a. DESCRIPTORS	b. IDENTIFIERS/OPEN ENDED TERMS	c. COSATI Field/Group
Air Pollution Dense Gas Mathematical Model Computer Model	Dispersion Elevated Sources	

18. DISTRIBUTION STATEMENT	19. SECURITY CLASS *(This Report)*	21. NO. OF PAGES 431
	20. SECURITY CLASS *(This page)*	22. PRICE

EPA Form 2220-1 (Rev. 4-77) PREVIOUS EDITION IS OBSOLETE

EPA-450/4-89-019

USER'S GUIDE
FOR THE DEGADIS 2.1
DENSE GAS DISPERSION MODEL

By

Tom Spicer
Jerry Havens

Office Of Air Quality Planning And Standards
Office Of Air And Radiation
U. S. Environmental Protection Agency
Research Triangle Park, NC 27711

November 1989

This report has been reviewed by the Office Of Air Quality Planning And Standards, U. S. Environmental Protection Agency, and has been approved for publication as received from the contractor. Approval does not signify that the contents necessarily reflect the views and policies of the Agency, neither does mention of trade names or commercial products constitute endorsement or recommendation for use.

EPA-450/4-89-019

Acknowledgements

This report was prepared by Jerry Havens, Department Of Chemical Engineering, University Of Arkansas, Fayetteville, AR 72701, under subcontract to PEI Associates, Cincinnati, OH 45246, in partial fulfillment of PEI's efforts under EPA Contract No. 68-02-4351. The EPA Project Officer for this report was Dave Guinnup, U. S. EPA (MD 14), Research Triangle Park, NC 27711. The report and computer code are being made available through the National Technical Information Service (NTIS), 5285 Port Royal Road, Springfield, VA 22161.

This page is intentionally left blank.

PREFACE

Version 2.1 of the elevated dense gas dispersion model, DEGADIS, has been developed by Dr. Jerry Havens and Dr. Tom Spicer of the University of Arkansas with the support of funding from the United States Environmental Protection Agency (EPA). While this model has not been extensively tested against field data and is subject to specific limitations and uncertainties, the EPA is making it publicly available through the National Technical Information Service (NTIS) as a research tool pending further model evaluation and development.

As developed, the DEGADIS 2.1 model is intended to completely replace the previous version of the model, DEGADIS 2.0. Version 2.1 incorporates a newly developed jet-plume model which will result in a plume which disperses in a fashion consistent with current Gaussian plume models if the plume becomes neutrally buoyant before it is predicted to fall to the ground. This represents a substantial improvement over the previous version and extends the applicability of the model into the not-denser-than-air regime. Specifically, the new jet-plume model provides for:

(a) automatic adjustment of integration step size using the Runge-Kutta-Gill method;
(b) elliptical plume cross-section with air entrainment specified consistent with the Pasquill-Gifford plume dispersion coefficient representation of atmospheric turbulent entrainment;
(c) user specification of averaging time;
(d) ground reflection of the plume when its lower boundary reaches ground level; and
(e) application to scenarios where the plume remains aloft.

A technical description of the new jet-plume model is found in Section III of this User's Guide.

Source code for the model is provided in archived form with a dearchiving program. The code may be dearchived on an IBM-compatible PC using the instructions listed in the file named "README.NOW". The dearchived source code files should then be transferred to a VAX computer. At this point several files must then be renamed prior to compilation and execution. Specific information on the renaming process is contained in the file AAREADME.TXT. Print this file and the compilation batch file, BUILD.COM, prior to attempting compilation.

It is the concern of the EPA that this model be applied only within the framework of its intended use. To this end the user is referred to the specific recommendations in Section VI for model application. In addition the EPA is in the process of preparing a general guidance document to assist the typical regulatory user in the execution of denser-than-air cloud dispersion models.

David E. Guinnup
Office of Air Quality Planning and Standards
U.S. Environmental Protection Agency

This page is intentionally left blank.

TABLE OF CONTENTS

Chapter	Page
List of Tables	ix
List of Figures	xi
List of Symbols	xiii
I. Introduction	1
II. Characterizing Gas Density Effects on Dispersion	2
III. Description of the Jet/Plume Model	5
IV. Evaluation of the Jet/Plume Model	17
V. Description of the DEGADIS Model	22
VI. Conclusions and Recommendations	49
References	51
Appendices	
A. Model Application on VAX/VMS Computers	A-1
B. Example Model Input and Output	B-1
C. Jet/Plume Model Source Code (JETPLU_IN and JETPLU_MAIN)	C-1
D. DEGADIS Model Source Code	D-1
E. JETPLU-DEGADIS Interface Source Code (DEGBRIDGE)	E-1
F. Partial Listing of Program Variables	F-1
G. DEGADIS Diagnostic Messages	G-1

This page is intentionally left blank.

LIST OF TABLES

Table		Page
II.1.	Criteria for Determining Dominance of Jet Effects for Ground-Level Releases	4
III.1.	Equations for a_e and b_e	6
III.2.	Constants for Determination of σ_z from Seinfeld (1986) after Turner (1969)	12
IV.1.	Specification of Gas Release Rates for Modeling	17
IV.2.	Comparison of Jet/Plume Model Prediction with Hoot et al.'s Wind Tunnel Data Correlation	19
IV.3.	Sensitivity of Jet/Plume Model Prediction to Variation of Entrainment Coefficients α_1 and α_2	21
V.1.	Typical Atmospheric Boundary Layer Stability and Wind Profile Correlations	40
V.2.	Values of $t_{a,i}$ for Use in Equation (V.101)	46
B.1.	MIC Example Simulation Release Conditions	B-1
B.2.	Ammonia Example Simulation Release Conditions	B-9
B.3.	Burro 9 Test Conditions	B-20

This page is intentionally left blank.

LIST OF FIGURES

Figure		Page
III.1.	Sketch of a Jet in a Developing Cross Flow	8
V.1.	Schematic Diagram of DEGADIS Dense Gas Dispersion Model	23
V.2.	Schematic Diagram of a Radially Spreading Cloud	25
V.3.	The Unsteady Gravity Current (van Ulden, 1983)	26
V.4.	The Head of a Steady Gravity Current (Simpson and Britter, 1979; van Ulden, 1983)	28
A.1.	Example DEGADIS Command Procedure on VAX/VMS for a Steady-State Simulation Named TEST_S	A-4
A.2.	Example DEGADIS Command Procedure on VAX/VMS for a Steady-State Simulation Named TEST	A-5
A.3.	RUN_NAME.IN Structure Required by JETPLU_IN	A-7
A.4.	DEGADISIN Flowchart	A-10
A.5.	Structure for Free-Formatted RUN_NAME.INP File	A-11
A.6.	DEGADIS1 Flowchart	A-12
A.7.	SYS$DEGADIS:EXAMPLE.ER1 Listing	A-15
A.8.	DEGADIS2 Flowchart	A-18
A.9.	SYS$DEGADIS:EXAMPLE.ER2 Listing	A-20
A.10.	DEGADIS3 Flowchart	A-22
A.11.	SYS$DEGADIS:EXAMPLE.ER3 Listing	A-24
A.12.	DEGADIS4 Flowchart	A-26
A.13.	Structure of Input for DEGADIS4	A-27
A.14.	SDEGADIS2 Flowchart	A-29
B.1.	Listing of EX1.IN	B-2
B.2.	Example Command File Used to Simulate the EX1 Simulation	B-2

LIST OF FIGURES (continued)

Figure		Page
B.3.	LIS File from the Output Files of JETPLU and DEGADIS	B-4
B.4.	Listing of EX2.IN	B-10
B.5.	Ammonia Aerosol/Air Adiabatic Mixture Density	B-11
B.6.	Ground-Level Isocontours--Oklahoma Ammonia Pipeline Break	B-11
B.7.	LIS File from Output Files of JETPLU and DEGADIS	B-12
B.8.	BURRO9S.INP Listing	B-29
B.9.	BURRO9.INP Listing	B-29

LIST OF SYMBOLS

a_e	empirical coefficient in Equation (III.2)
a_v	empirical constant (1.3) in Equations (V.6) and (V.27)
B_{EFF}	effective width of gas plume (m)
B_i'	local half width of source seen by observer i (m)
b	half width of horizontally homogeneous central section of gas plume (m)
b_e	empirical power in Equation (III.2)
b_j	characteristic jet width (m)
b_v	empirical constant (1.2) in Equations (V.8) and (V.28)
C_E	constant (1.15) in density intrusion (spreading) relation
C_p	heat capacity (J/kg K)
C_{p_a}	heat capacity of air (J/kg K)
C_{p_c}	heat capacity of contaminant (J/kg K)
C_{p_w}	heat capacity of water (liquid phase) (J/kg K)
c	local concentration (kg/m^3)
c_c	centerline concentration (kg/m^3)
$c_{c,L}$	vertically averaged layer concentration (kg/m^3)
c_f	friction coefficient
c_c'	centerline, ground-level concentration corrected for x-direction dispersion (kg/m^3)
D	source diameter (m)
D_h	added enthalpy (J/kg)
D'	diffusivity (m^2/s)
d_v	empirical constant (0.64) in Equations (V.4) and (V.12)
E	plume strength (kg/s)

E(t)	contaminant primary source rate (kg contaminant/s)
E_1	air entrainment associated with a free turbulent jet (kg/ms)
E_2	air entrainment associated with a bending plume (kg/ms)
E_3	jet/plume air entrainment associated with passive atmospheric dispersion (kg/ms)
$E_{3,a}$	air entrainment associated with passive atmospheric dispersion (kg/ms)
e_v	empirical constant (20.) in Equations (V.5) and (V.15)
F	overall mass transfer coefficient (kg/m^2 s)
F_f	mass transfer coefficient due to forced convection (kg/m^2 s)
F_n	mass transfer coefficient due to natural convection (kg/m^2 s)
Fr	Froude number, $\rho_a u_a^2/(g(\rho_e - \rho_a)D)$
Fr_h	"horizontal" Froude number, $u_a/[gD(\rho_e - \rho_a)/\rho_a]^{1/2}$
Fr_r	"release" Froude number, $V/[gD(\rho_e - \rho_a)/\rho_a]^{1/2}$
Gr	Grashoff number
g	acceleration of gravity (m/s^2)
H	characteristic release depth or depth of density intrusion or cloud (m)
H_a	ambient absolute humidity (kg water/kg dry air)
H_{EFF}	effective cloud depth (m)
H_h	height of head in density-driven flow (m)
H_L	total layer depth (m)
H_r	maximum plume rise above stack (m)
H_s	stack (exit) height (m)
H_t	height of tail in density-driven flow (m)
H_1	average depth of gravity current head (m)
H_4	depth of inward internal flow in a gravity current head (m)

h	enthalpy of source blanket (J/kg)
h_a	enthalpy of ambient humid air (J/kg)
h_E	enthalpy associated with primary source mass rate (J/kg)
h_f	heat transfer coefficient due to forced convection (J/m^2 s K)
h_L	enthalpy of vertically averaged layer (J/kg)
h_n	heat transfer coefficient due to natural convection (J/m^2 s K)
h_0	overall heat transfer coefficient (J/m^2 s K)
h_p	enthalpy of primary source (J/kg)
h_w	enthalpy associated with mass flux of water from surface (J/kg)
J	release momentum ratio
K_0	constant in Equation (V.96) ($m^{1-\gamma_1}$)
K_y	horizontal turbulent diffusivity (m^2/s)
K_z	vertical turbulent diffusivity (m^2/s)
k	von Karman's constant, 0.35
$k_1 - k_6$	constants defined by Equations (III.14)-(III.19)
L	source length (m)
L_B	buoyancy length scale (m)
M	total cloud mass (kg)
M_a	total mass of air in the cloud (kg)
M_c	total mass of contaminant in the cloud (kg)
M_i	initial cloud mass (kg)
MW	molecular weight
$\dot{M}_a$	mass rate of air entrainment into the cloud (kg/s)
$\dot{M}_{w,s}$	mass rate of water transfer to the cloud from the water surface under the source (kg/s)

N	number of observers
Nu	Nusselt number
P	cloud momentum (kg m/s)
P_e	perimeter of R' (m)
P_h	momentum of head in density-driven flow (kg m/s)
P_t	momentum of tail in density-driven flow (kg m/s)
P_v	virtual momentum due to acceleration reaction (kg m/s)
Pr	Prandtl number
p	atmospheric pressure (atm)
$p_{w,c}$	partial pressure of water in the cloud (atm)
$p^*_{w,s}$	vapor pressure of water at the surface temperature (atm)
Q	volumetric release rate (m^3/s)
Q_E	source mass flux (kg/m^2 s)
Q_e	volumetric entrainment flux (m/s)
Q_1	flux of ambient fluid into front of gravity current head (m/s)
$\dot{Q}_s$	rate of heat transfer from the surface (J/s)
Q_*	atmospheric takeup flux (kg/m^2 s)
Q_{*max}	maximum potential atmospheric takeup flux of contaminant (kg/m^2 s)
q_s	surface heat flux (J/m^2 s)
R	gas source radius (m)
R_h	inner radius of head in density-driven flow (m)
R_m	value of R when ($\pi R^2 Q_*$) is a maximum (m)
R_{max}	maximum radius of the cloud (m)
R_p	primary source radius (m)
R'	jet/plume region perpendicular to the axis

Ri_c	release Richardson number
Ri_f	Richardson number associated with the front velocity, Equation (V.11)
Ri_T	Richardson number associated with temperature differences, Equation (V.88)
Ri'_*	Richardson number associated with density differences corrected for convective scale velocity
Ri_*	Richardson number associated with density differences, Equation (V.78)
r	radial distance to jet/plume axis (m)
S	length of zone of flow establishment (m)
Sc	Schmidt number
Sh	Sherwood number
St_H	Stanton number for heat transfer
St_M	Stanton number for mass transfer
S_y	horizontal concentration scaling parameter (m)
S_z	vertical concentration scaling parameter (m)
S_{z0}	S_z at the downwind edge of the source ($x = L/2$) (m)
S_{z0_m}	value of S_{z0} when $(\pi R^2 Q_*)$ is a maximum (m)
s	distance along plume axis (m)
T	temperature associated with source blanket enthalpy (K)
$T_{c,L}$	temperature associated with layer-averaged enthalpy (K)
T_s	surface temperature (K)
t	time (s)
t_a	averaging time (s)
$t_{a,i}$	"instantaneous" averaging time associated with puff dispersion coefficients
t_p	averaging time associated with peak measured concentrations (s)

t_s	specified time (s)
t_{dn_i}	time when observer i encounters downwind edge (s)
t_{up_i}	time when observer i encounters upwind edge (s)
u	ambient (wind) velocity (m/s)
u_a	ambient average velocity (m/s)
u_c	jet/plume excess velocity at axis (m)
u_e	horizontal or frontal entrainment velocity (m/s)
u_{EFF}	effective cloud advection velocity (m/s)
u_f	cloud front velocity (m/s)
u_i	velocity of observer i (m/s)
u_L	average transport velocity associated with H_L (m/s)
u_x	wind velocity, along x-direction (m/s)
u_0	wind velocity measured at $z = z_0$ (m/s)
u_3	internal flow out of gravity current head (m/s)
u_4	internal flow into gravity current head (m/s)
u_*	friction velocity (m/s)
$\bar{u}$	characteristic average velocity (m/s)
V	jet velocity (m/s)
V_H	heat transfer velocity (0.0125 m/s) in Equation (V.38) (m/s)
w_a	mass fraction of air
w_c	mass fraction of contaminant
$w_{c,p}$	mass fraction of contaminant in primary source
w_e	vertical entrainment velocity associated with H_L (m/s)
w_*	convective scale velocity (m/s)
w'_e	entrainment velocity associated with H_{EFF} (m/s)
x_c	downwind distance to centerline ground contact (m)

$x_i(t)$	x position of observer i at time t (m)
x_{pi}	position of puff center due to observer i (m)
x_t	downwind distance where gravity spreading terminates (m)
x_v	virtual point source distance (m)
x_{dn_i}	x position of downwind edge of source for observer i
x_{up_i}	x position of upwind edge of source for observer i
$\bar{x}$	downwind distance to maximum rise (m)
x,y,z	Cartesian coordinates (m)
x_0	downwind edge of the gas source (m)
y'	local lateral dimension (m)
z_j	elevation of jet/plume centerline (m)
z_R	surface roughness (m)
z_0	reference height in wind velocity profile specification (m)
z'	local vertical dimension perpendicular to the jet/plume axis (m)
α	constant in power law wind profile
α_1	jet entrainment coefficient
α_2	line thermal entrainment coefficient
β_y	constant in σ_y correlation in Equation (V.97)
β_z	power in σ_z correlation
Γ	gamma function
γ	ratio of $(\rho - \rho_a)/c$
γ_1	constant in Equation (V.96)
γ_z	constant in σ_z correlation
Δ	ratio of $(\rho - \rho_a)/\rho$
ΔT	temperature driving force (K) $(T_s - T_{c,L})$ or $(T_s - T)$

Δ'	ratio of $(\rho - \rho_a)/\rho_a$
δ	empirical constant (2.15)
δ_L	empirical constant (2.15) in Equation (V.53)
δ_v	constant (0.20) in Equation (V.25)
δ_y	constant in σ_y correlation in Equation (V.97)
δ_z	constant in σ_z correlation
ϵ	frontal entrainment coefficient (0.59) in Equation (V.33)
ς	collection of terms defined by Equation (V.62) ($m^{-1/(1+\alpha)}$)
λ	Monin-Obukhov length (m)
μ	viscosity (kg/m s)
ρ	density of gas-air mixture (kg/m^3)
ρ_a	air density (kg/m^3)
ρ_c	jet/plume excess density at axis (kg/m^3)
ρ_e	density of released gas (kg/m^3)
ρ_L	vertically averaged layer density (kg/m^3)
ρ_0	density of contaminant's saturated vapor at T_0 (kg/m^3)
θ	angle between plume axis and horizontal (radians)
λ^2	turbulent Schmidt number, 1.42
σ_r	concentration profile parameter for an axisymmetrical jet/plume (m)
$\sigma_{x,a}$	Pasquill-Gifford x-direction (ambient) dispersion coefficient (m)
$\sigma_{y,a}$	Pasquill-Gifford y-direction (ambient) dispersion coefficient (m)
$\sigma_{y'}$	concentration profile parameter in the y' direction (m)
$\sigma_{z,a}$	Pasquill-Gifford z-direction (ambient) dispersion coefficient (m)
$\sigma_{z'}$	concentration profile parameter in the z' direction (m)
σ_1^2	variance associated with E_1 (m^2)
σ_2^2	variance associated with E_2 (m^2)

ϕ	function describing influence of stable density stratification on vertical diffusion, Equation (V.76)
$\hat{\phi}$	integrated source entrainment function
Ψ	logarithmic velocity profile correction function
χ_c	jet centerline concentration at centerline touchdown

(This page is intentionally left blank.)

I. INTRODUCTION

Episodic releases of hazardous chemical gases from chemical process pressure relief operations can pose significant hazards to public health. Conventional air pollutant dispersion models may not be applicable for assessing the consequences of such releases, particularly when the gases released are denser than air. The DEGADIS model (Havens and Spicer, 1985) was designed to model the atmospheric dispersion of ground-level, area-source dense gas (or aerosol) clouds released with zero (initial) momentum into an atmospheric boundary layer over flat, level terrain. DEGADIS describes the dispersion processes which accompany the ensuing gravity-driven flow and entrainment of the gas into the atmospheric boundary layer, and it has been verified by comparison with a wide range of laboratory and field-scale heavy gas release/dispersion data. However, DEGADIS made no provision for processes which occur in high velocity releases, as from pressure relief valves.

Ooms, Mahieu, and Zelis (1974) reported a mathematical model for estimating the trajectory and dilution of dense gas jet plumes. The model comprised simplified balance equations for mass, momentum, and energy, with Gaussian similarity profiles for velocity, density, and concentration in the jet. Havens (1988) interfaced Ooms' jet-plume model with DEGADIS to provide for prediction of the trajectory and dilution of an elevated dense gas jet to ground contact, with (DEGADIS) prediction of the ensuing ground-level plume dispersion.

The purpose of this work was to improve the jet-plume/DEGADIS model as follows:

- The jet-plume part of the model was modified to provide for elliptical plume shape (cross-section) with air entrainment specified to be consistent with the Pasquill-Gifford plume dispersion coefficient representation of atmospheric turbulent entrainment. This modification allows application of the model to scenarios where the plume remains aloft.

- The jet-plume part of the model was modified to incorporate ground reflection when the plume (lower) boundary reaches ground level.

- The jet-plume part of the model was modified to provide for automatic adjustment of integration step-size (using the Runge-Kutta-Gill method as in DEGADIS). This modification improves the computational efficiency of the model.

II. CHARACTERIZING GAS DENSITY EFFECTS ON ATMOSPHERIC DISPERSION

Atmospheric dispersion of gas releases may involve the following fluid flow regimes:

jet • buoyancy-dominated • stably-stratified • passive dispersion

All four regimes, which may be present in different degrees depending on the rate and (characteristic) dimensions of the release, gas density, and characteristics of the atmospheric flow, should be accounted for in a general application dispersion model. The dispersion (and trajectory) of vertical jets (perpendicular to the wind flow) can be modeled with the jet-plume model described here, and the buoyancy-dominated, stably-stratified, and passive dispersion regimes (on level, unobstructed terrain) can be modeled with DEGADIS (Havens and Spicer, 1985).

The jet-plume model currently does not provide for simulation of horizontal jets (such as might occur from ruptured gas transfer lines). Consequently, we consider here other methods for estimating the relative importance of the four dispersion regimes for ground level releases with horizontal momentum. In rapid releases of large quantities of dense gas (with low initial momentum) a cloud having similar vertical and horizontal dimensions may form. In this "buoyancy-dominated regime", (gravity-induced) slumping and lateral spreading motion ensues until the kinetic energy of the buoyancy-driven flow is dissipated. The gravity-induced flow, which may effect mixing (primarily at the advancing vapor cloud front), can be an important determinant of the shape and extent of the gas cloud. After the kinetic energy of the buoyancy-driven flow is dissipated, the dispersion process following can be described as a "stably stratified" plume (or cloud) embedded in the mean wind flow. The density stratification present in this regime, which can be much stronger than that occurring naturally in the atmospheric boundary layer, tends to damp turbulence and reduce vertical mixing. As the dispersion proceeds, the stable stratification due to the dense gas decreases until the dispersion process can be represented as a neutrally buoyant plume (or cloud) in the mean wind flow. This (final) dispersion regime can be predicted with passive contaminant dispersion theory.

Havens and Spicer (1985) suggested criteria (based on water tunnel experiments reported by Britter (1980)) for determining the importance of each of the low-momentum flow regimes described above. In Britter's experiments brine was released (at floor level) into a water tunnel flow, and the lateral and upwind extent of the brine/water plume was measured as a function of the buoyancy length scale

$$L_B = Qg(\rho e - \rho_a)/(\rho_e u^3)$$

where Q was the volumetric (brine) emission rate and u was the water-tunnel flow velocity. Britter's data indicated releases were passive from the source when $L_B/D \lesssim 0.005$ and were dominated by the negative buoyancy dispersion regime when $L_B/D \gtrsim 0.1$. The following Release Richardson number criteria, based on these observations, were suggested:

If $Ri_C \gtrsim 30$ negative buoyancy-dominated

If $1 \lesssim Ri_c \lesssim 30$ stably stratified shear flow

If $Ri_c \lesssim 1$ passive dispersion (II.1)

where $Ri_c = g(\rho_e - \rho_a)H/(\rho_a u_*)$. The reported values reflect the ratio $(u/u_*) = 16$ for Britter's water-tunnel flow, and the length scale corresponding to the depth of the layer was approximated by $H = Q/uD$.

High (initial) momentum releases require additional consideration if jet entrainment dominates other air entrainment mechanisms. In the negative buoyancy-dominated regime, the ambient flow does not strongly affect the rate of air entrainment. Consequently, the relative importance of buoyancy-driven and (horizontal, near-ground-level) jet flow effects can be evaluated using the criterion based on Britter's data (with the release velocity V used in place of the ambient velocity in the buoyancy length scale). Jet entrainment dominates the negative buoyancy regime entrainment when

$$\left[\frac{V}{u_*}\right]^2 \gtrsim 10\ Ri_c \tag{II.2}$$

This criterion suggests that dominance of jet effects decreases with increased jet density (since buoyancy-driven flow frontal air entrainment increases).

The air entrainment into a turbulent free jet can be estimated as (Wheatley, 1986):

$$\frac{dM_a}{dx} = \left[\frac{0.159}{2}\right]\left[\frac{2\pi R}{2}\right] V \rho_a \tag{II.3}$$

The rate of (vertical) air entrainment (per unit width) for the stably-stratified shear flow and passive dispersion regimes predicted by DEGADIS is

$$\frac{d}{dx}(\rho_L u_L H_L) = \frac{\rho_a \delta_L k u_*(1+\alpha)}{\phi(Ri_*)} \tag{II.4}$$

where $\phi(Ri_*) \cong 0.88 + 0.099\ Ri_*$. Using an effective width of $2\pi R$, a typical value of $\alpha = 0.2$, $\delta_L = 2.1$, $k = 0.35$, and $(u/u_*) = 30$ (typical of atmospheric boundary layers), the above equations can be combined to show that jet entrainment dominates the stably-stratified flow regime entrainment when

$$\frac{V}{u} \gtrsim 16/(19 + Ri_c) \qquad \text{(II.5)}$$

This criterion suggests that dominance of jet effects increases with increased jet density (since stably-stratified flow entrainment decreases). Further, when $Ri_c \lesssim 1$ the criterion suggests that jet effects dominate the passive dispersion regime when $(V/u) \gtrsim 0.8$, which is consistent with the criterion suggested by Cude (1974) and Wheatley (1986).

Summarizing, the following procedure can be used to determine the (starting) dominant dispersion regime:

(1) Calculate $Ri_c = g(\rho_e - \rho_a)H/(\rho_a u_*^2)$.

(2) Determine the dominant (nonjet) dispersion regime using Equation (II.1).

(3) Determine if horizontal jet effects dominate the regime determined in (2), using the relationships summarized in Table II.1.

Table II.1. Criteria for Determining Dominance of Jet Effects for Ground-Level Releases

Ground-level jet effects dominate:	
• negative buoyancy-dominated regimes when	$\left(\frac{V}{u_*}\right)^2 \gtrsim 10\ Ri_c$,
• stably stratified shear flow regimes when	$V/u \gtrsim 16/(19 + Ri_c)$, and
• passive dispersion regimes when	$V/u \gtrsim 0.8$

III. DESCRIPTION OF THE JET-PLUME MODEL

Gas (or aerosol) jet releases, such as might occur in chemical process pressure relief operations, are modeled as vertically directed releases of pure material into the atmospheric flow field. The jet is assumed to be discharged from a circular vent with a (constant) uniform velocity profile. At some distance downwind of the release, the velocity and concentration profiles are assumed Gaussian.

Zone of Flow Establishment

The region before Gaussian similarity velocity and concentration profiles are established is the so-called zone of flow establishment. Pratte and Baines (1967) reported that the length of the potential core (S_o) of a jet released from a tube perpendicular to a cross flow approaches the potential core length for a turbulent free jet with no cross flow as the ratio of the jet velocity to the cross-flow velocity (V/u_a) becomes large. Although S_o is also a function of the Reynolds number (in the tube), Reynolds number dependence is less important for large release momentum. Pratte and Baines' (largest Reynolds number) data were correlated here as

$$(S_o/D) = 6.4\,(1 - \exp(-0.48(V/u_a))) \tag{III-1}$$

where D is the diameter of the release. The form of Equation (III-1) was chosen so that (S_o/D) approaches 0 as (V/u_a) approaches 0 and (S_o/D) approaches 6.4 as (V/u_a) becomes large. Chassaing et al. (1974) point out that similarity profiles are not established until some distance after the potential core has vanished. Consequently, the length of the zone of flow establishment (S) is modeled as

$$(S/D) = 7.7 \left\{ 1 - \exp\left(-0.48\,\frac{\rho_e V}{\rho_a u_a} \right) \right\} \tag{III.2}$$

where (V/u_a) has been replaced with ($\rho_e V/\rho_a u_a$) to allow extended application for non-ambient-density gases.

Kamotani and Greber (1972) parameterized the trajectory of the jet-plume velocity profile in the zone of flow establishment as

$$x/D = a_e\,(z/D)^{b_e} \tag{III.3}$$

where a_e and b_e are empirical constants determined as a function of the momentum ratio of the release ($J = \rho_e V^2/(\rho_a u_a^2)$). Astleford et al. (1983) determined equations for a_e and b_e as functions of J from Kamotani and Greber's data. Morrow (1985) revised the equations to extend the range for smaller values of J and to include the effect of a release Froude number ($Fr = \rho_a u_a^2/(g(\rho_e - \rho_a)D)$); these equations are shown in Table III.1. Correlations were also determined here for a_e and

b_e for $50 < J < 600$ from Kamotani and Greber's original work, based on a least squares fit; these equations are also assumed to apply for $J > 600$.

Table III.1. Equations for a_e and b_e.

Momentum Ratio (J) Range	Froude Number (Fr) Range	a_e	b_e	Ref.
$J \leq 0.036$	all	$18.519 J$	0.4	1
$0.036 < J \leq 10$	$Fr > 1.688$ or $Fr \leq 0$	$\exp[0.405465+0.24386 \ln J]$	0.4	1
	$0 < Fr \leq 1.688$	$\exp[0.2476+0.3016 \ln Fr +0.24386 \ln J)]$	0.4	1
$10 < J \leq 50$	all	$\exp[0.405465+0.131386 \ln J +0.054931 (\ln J)^2]$	$\exp[-0.744691 -0.074525 \ln J]$	2
$50 < J \leq 600$	all	$\exp[-2.58012+1.49202 \ln J -0.097623 (\ln J)^2]$	$\exp[-0.446718 -0.1950694 \ln J]$	3

References:
1 Morrow (1985)
2 Astleford, Morrow, and Buckingham (1983)
3 this work

At the end of the zone of flow establishment, the maximum jet-plume velocity is (approximately) the release velocity (V), and the centerline concentration is (approximately) the pure contaminant concentration. The jet angle (with the horizontal) is determined by x, z, and S from Equations (III.2) and (III.3).

Zone of Established Flow

For the zone of established flow, Ooms (1972) assumed Gaussian similarity profiles (single length scale) for velocity, density, and concentration. The concentration profile was

$$c(r,s) = c_c(s)\ \exp\left\{ - \left[\frac{r}{\lambda\ b_j(s)} \right]^2 \right\} \qquad \text{(III.4)}$$

where c_c represents the centerline concentration and λ^2 is a turbulent Schmidt number which represents the square of the ratio of the characteristic length of the concentration profile to the characteristic length of the velocity profile (b_j). By contrast, the Pasquill-Gifford representation of atmospheric turbulent dispersion assumes the following Gaussian profile:

$$c(x,y,z) = c_c(x)\ \exp\left\{ - \frac{1}{2} \left[\frac{y}{\sigma_y(x)} \right]^2 - \frac{1}{2} \left[\frac{z}{\sigma_z(x)} \right]^2 \right\} \qquad \text{(III.5)}$$

For a symmetric plume, $\sigma_y = \sigma_z$ ($=\sigma_r$), and $r^2 = y^2 + z^2$; Equation (III.5) becomes

$$c(x,r) = c_c(x)\ \exp\left\{ - \frac{1}{2} \left[\frac{r}{\sigma_r(x)} \right]^2 \right\}, \qquad \text{(III.6)}$$

and the profile,

$$c(s,y',z') = c_c(s)\ \exp\left\{ - \frac{1}{2} \left[\frac{y'}{\sigma_{y'}(s)} \right]^2 - \frac{1}{2} \left[\frac{z'}{\sigma_{z'}(s)} \right]^2 \right\} \qquad \text{(III.7)}$$

where y' and z' denote the local lateral and vertical dimensions in the jet-plume (Figure III.1), incorporates Equations (III.4) and (III.5) and provides for a circular cross section initially ($\sigma_{y'} = \sigma_{z'}$) with transition to an elliptical cross section.

Associated profiles for velocity and density are also assumed:

$$u(s,y',z') = u_a \cos\theta + u_c(s)\ \exp\left\{ - \frac{1}{2} \left[\frac{\lambda\ y'}{\sigma_{y'}(s)} \right]^2 - \frac{1}{2} \left[\frac{\lambda\ z'}{\sigma_{z'}(s)} \right]^2 \right\} \qquad \text{(III.8)}$$

$$\rho(s,y',z') = \rho_a + \rho_c(s)\ \exp\left\{ - \frac{1}{2} \left[\frac{y'}{\sigma_{y'}(s)} \right]^2 - \frac{1}{2} \left[\frac{z'}{\sigma_{z'}(s)} \right]^2 \right\} \qquad \text{(III.9)}$$

Note that the turbulent Schmidt number now appears in the velocity profile. $\sigma_{y'}$ and $\sigma_{z'}$ should approach $\sigma_{y,a}$ and $\sigma_{z,a}$ for passive atmospheric dispersion as the density and momentum of the jet/plume approach ambient values.

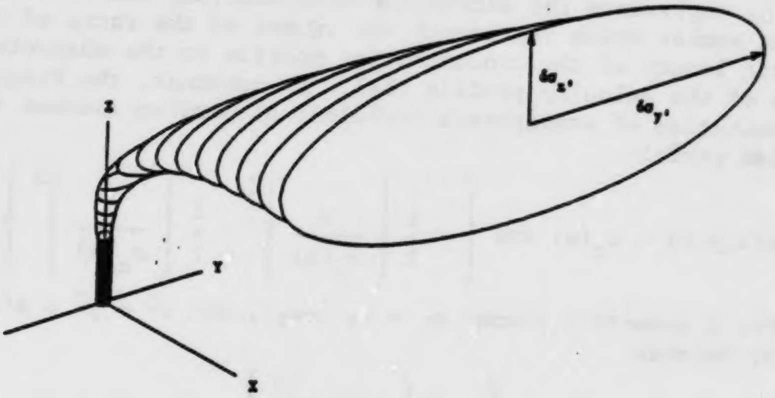


Figure III.1. Sketch of a Developing Jet in a Cross Flow.

The fundamental balance equations are:

Contaminant Mass Balance

$$\frac{d}{ds}\int_{R'} cu\, dR' = 0 \tag{III.10}$$

Total Mass Balance

$$\frac{d}{ds}\int_{R'} \rho u\, dR' = E_1 + E_2 + E_3 \tag{III.11}$$

X-Direction Momentum Balance

$$\frac{d}{ds}\int_{R'} \rho u\,(u \cos\theta)\, dR' = u_a(E_1 + E_2 + E_3) + c_d P_e \rho_a u_a^2 |\sin^3\theta|/2 \tag{III.12}$$

Z-Direction Momentum Balance

$$\frac{d}{ds}\int_{R'} \rho u\,(u \sin\theta)\, dR' = \int_{R'} g(\rho_a - \rho)\, dR' - \text{sign}(\theta)\, c_d P_e \rho_a u_a^2 \sin^2\theta \cos\theta/2 \tag{III.13}$$

The jet/plume boundary is assumed located where the concentration is one-tenth of the centerline value and encloses the region R'. (R' is an ellipse or circle with major axes $\delta\sigma_{y'}$ and $\delta\sigma_{z'}$ where δ is a constant equal to 2.15; P_e is the perimeter of R'.) The term $(E_1 + E_2 + E_3)$ represents air entrainment into the jet plume by the three (independent) mechanisms discussed below. A drag force term assumed to act at right angles to the jet-plume trajectory is included on the right hand sides of Equations (III.12) and (III.13). The value used for c_d is 0.2; although this value of c_d is different from the value used by Ooms (1972) (since different length scales were used), its use in Equations (III.12) and (III.13) provides drag forces consistent with Ooms' predictions.

A (thermal) energy balance is not included in the balance equations given above. In the original formulation and in later work (Ooms, Mahieu, and Zelis, 1974), the energy balance was used to determine the developing jet temperature. However, since no provision was made for thermal energy transfer between the jet and its surroundings (e.g., the ground), the contaminant was (implicitly) assumed to mix adiabatically with ambient air, and an energy balance is not required if the adiabatic mixing properties are calculated beforehand.

For adiabatic mixing of two ideal gases with equal constant molal heat capacities, the ratio $\gamma = (\rho - \rho_a)/c$ is constant. Assumption of the density profile in Equation (III.9) and the concentration profile in Equation (III.7) is equivalent to assuming γ constant at any given distance, s. In the model development, $d\gamma/ds$ was assumed to be negligible. In the model implementation, γ is updated with distance based on the adiabatic mixing properties.

In Equations (III.10) - (III.13), u_a is considered constant for evaluation of the integrals. The value of u_a used is the averaged velocity over the top half of the jet plume, and u_a is updated with distance.

The integrals on the left-hand sides of Equations (III.10) - (III.13) do not have closed form solutions when R' represents an ellipse. If R' is approximated by a rectangle with an area equal to that of the original ellipse, the integrations are readily performed. (The approximating rectangle has sides with length $\sqrt{\pi}\delta\sigma_{y'}$ and $\sqrt{\pi}\delta\sigma_{z'}$.) The constants in Ooms' original formulation were compared to constants determined by approximating the circular cross section as square; the difference was less than 2%.

With the approximations stated above, six constants appear in the resulting equations:

$$k_1 = \frac{1}{\sigma_{y'}\sigma_{z'}} \int_{R'} \exp\left\{ -\frac{1}{2}\left[\left[\frac{y'}{\sigma_{y'}}\right]^2 + \left[\frac{z'}{\sigma_{z'}}\right]^2\right]\right\} dR' \qquad \text{(III.14)}$$

$$k_2 = \frac{1}{\sigma_{y'}\sigma_{z'}} \int_{R'} \exp\left\{ -\frac{(1+\lambda^2)}{2} \left[\left[\frac{y'}{\sigma_{y'}}\right]^2 + \left[\frac{z'}{\sigma_{z'}}\right]^2 \right] \right\} dR' \qquad \text{(III.15)}$$

$$k_3 = \frac{1}{\sigma_{y'}\sigma_{z'}} \int_{R'} dR' = \pi \, \delta^2 \qquad \text{(III.16)}$$

$$k_4 = \frac{1}{\sigma_{y'}\sigma_{z'}} \int_{R'} \exp\left\{ -\frac{\lambda^2}{2} \left[\left[\frac{y'}{\sigma_{y'}}\right]^2 + \left[\frac{z'}{\sigma_{z'}}\right]^2 \right] \right\} dR' \qquad \text{(III.17)}$$

$$k_5 = \frac{1}{\sigma_{y'}\sigma_{z'}} \int_{R'} \exp\left\{ -\lambda^2 \left[\left[\frac{y'}{\sigma_{y'}}\right]^2 + \left[\frac{z'}{\sigma_{z'}}\right]^2 \right] \right\} dR' \qquad \text{(III.18)}$$

$$k_6 = \frac{1}{\sigma_{y'}\sigma_{z'}} \int_{R'} \exp\left\{ -\frac{(1+2\lambda^2)}{2} \left[\left[\frac{y'}{\sigma_{y'}}\right]^2 + \left[\frac{z'}{\sigma_{z'}}\right]^2 \right] \right\} dR' \qquad \text{(III.19)}$$

Air Entrainment

The balance equations for mass and x-direction momentum require specification of the air entrainment rate (E_1, E_2, and E_3 in Equations (III.11) and (III.13)).

E_1 represents the (near-field) air entrainment associated with jetting of the release. Although a cross flow is present, the momentum of the release dominates the ambient momentum when jetting is important, and the region near the jet release can be modeled as a free turbulent jet:

$$E_1 = \alpha_1 \rho_a P_e u_c \qquad \text{(III.20)}$$

where P_e is the perimeter of R' and α_1 is the entrainment coefficient associated with this area. Note that E_1 is taken to be zero if $u_c < 0$. A value of $\alpha_1 = 0.028$ is based on a summary of free turbulent jet experiments reported by List (1982); although this value of α_1 is different from the value used by Ooms (1972) (since different length scales were used), its use in Equation (III.20) provides rates of air entrainment consistent with Ooms' predictions.

E_2 represents the air entrainment associated with a cross-flow velocity component perpendicular to the jet-plume centerline. In the absence of jetting, large scale vortices (which entrain air) will form in a plume due to the cross flow. Richards (1963) measured the air entrainment rate into line thermals released in a calm environment. Based on Richards' experiments, E_2 is modeled as follows:

$$E_2 = \alpha_2 \rho_a P_e u_a |\sin\theta| \cos\theta \tag{III.21}$$

where $u_a|\sin\theta|$ is the velocity component perpendicular to the jet-plume centerline and $\alpha_2 = 0.37$ is based on Richards' data. Ooms (1972) added the $\cos\theta$ term so that this contribution is present only in the far field. Again, although this value of α_2 is different from the value used by Ooms (1972), its use in Equation (III.21) provides rates of air entrainment consistent with Ooms' predictions.

E_3 represents the atmospheric air entrainment for a passive plume. For a passive plume, the atmospheric air entrainment rate ($E_{3,a}$) is given by

$$E_{3,a} = \frac{d}{dx}\int_{R'} \rho_a u_a \, dR' = \rho_a u_a \frac{d}{dx}\int_{R'} dR' = \rho_a u_a \frac{d}{dx}\left[k_3 \sigma_{y,a} \sigma_{z,a} \right] \tag{III-22}$$

where $\sigma_{y,a}$ and $\sigma_{z,a}$ denote the ambient lateral and vertical Pasquill-Gfford dispersion coefficients, respectively. Since k_3 is constant, Equation (III.22) becomes

$$E_{3,a} = k_3 \rho_a u_a \left[\sigma_{z,a} \frac{d\sigma_{y,a}}{dx} + \sigma_{y,a} \frac{d\sigma_{z,a}}{dx} \right] \tag{III.23}$$

where the derivatives can be calculated from accepted correlations for $\sigma_{y,a}$ and $\sigma_{z,a}$, such as

$$\sigma_{y,a} = \delta_y x^{\beta_y} \tag{III.24}$$

$$\sigma_{z,a} = \delta_z x^{\beta_z} \exp(\gamma_z (\ln(x))^2) \tag{III.25}$$

(Seinfeld, 1986). Typical values for δ_z, β_z, and γ_z are given in Table III.2.

Methods recommended in Section V for determining the effect of averaging time are applied for the jet-plume model, and values for δ_y and β_y as a function of stability and averaging time are given in Equation (V.101). For the jet plume, E_3 is modeled as

$$E_3 = k_3 \rho_a u_a \left[\sigma_{z,a} \frac{d\sigma_{y,a}}{dx} + \sigma_{z,a} \frac{d\sigma_{y,a}}{dx} \right] \cos\theta \tag{III.26}$$

where the $\cos\theta$ term is included because E_3 is required as a function of s.

Table III.2. Constants for Determination of σ_z from Seinfeld (1986) after Turner (1970)

Stability Class	δ_z[1]	β_z	γ_z
A	107.7	-1.7172	0.2770
B	0.1355	0.8752	0.0136
C	0.09623	0.9477	-0.0020
D	0.04134	1.1737	-0.0316
E	0.02275	1.3010	-0.0450
F	0.01122	1.4024	-0.0540

[1] for use in $\sigma_z = \delta_z x^{\beta_z} \exp(\gamma_z(\ln(x))^2)$ where σ_z and x are in meters.

Gravitational Force

The gravitational force term in Equation (III.13) is given by

$$\int_{R'} g(\rho_a - \rho)\, dR' = g\gamma \int_{R'} c\, dR' = k_1 g\gamma c_c \sigma_{y'} \sigma_{z'} \qquad \text{(III.27)}$$

As discussed earlier, γ is assumed constant in the equation development but is updated with distance in the model implementation. Note that Equation (III.27) applies for $\rho \le \rho_a$ as well as $\rho > \rho_a$.

Additional Constraints

Along with an equation of state (such as the ideal gas law or an adiabatic mixing relationship between concentration and density), balance Equations (III.10) - (III.13) provide five constraints on the six unknowns c_c, ρ_c (or γ), u_c, θ, $\sigma_{y'}$, and $\sigma_{z'}$. Therefore, an additional constraint is required to solve the equations. The momentum balances provide the information needed to determine θ and u_c. The contaminant mass balance provides the information to determine c_c. The total mass balance provides the information to determine the product $\sigma_{y'}\sigma_{z'}$, but the balance equations provide no guidance for the individual values of $\sigma_{y'}$ and $\sigma_{z'}$. (No additional constraint was needed in Ooms' model since a single length scale was used.) The last constraint is to be specified so that the values of $\sigma_{y'}$ and $\sigma_{z'}$ are consistent with atmospheric processes.

The model treats three (assumed) independent mechanisms for air entrainment. For independent processes, the variances are additive:

$$\sigma_{y'}^2 = \sigma_1^2 + \sigma_2^2 + \sigma_{y,a}^2 \qquad \text{(III.28)}$$

$$\sigma_{z'}^2 = \sigma_1^2 + \sigma_2^2 + \sigma_{z,a}^2 \qquad \text{(III.29)}$$

where σ_1^2 and σ_2^2 represent the variances associated with entrainment processes E_1 and E_2, respectively. Combining Equations (III.28) and (III.29):

$$\sigma_{y'}^2 - \sigma_{z'}^2 = \sigma_{y,a}^2 - \sigma_{z,a}^2 \qquad \text{(III.30)}$$

Equation (III.30) acts as the final constraint on the system. In the near field, $\sigma_{y,a}^2 - \sigma_{z,a}^2$ is small and $\sigma_{y'} = \sigma_{z'}$. In the far field, $\sigma_{y'}$ and $\sigma_{z'}$ will approach $\sigma_{y,a}$ and $\sigma_{z,a}$, respectively.

Closure

Initial conditions for the zone of established flow are determined from the zone of flow establishment; initial values of $\sigma_{y'}$ and $\sigma_{z'}$ are determined from the contaminant mass balance and the (known) release rate. With $\lambda^2 = 1.42$ (List, 1982), the assumed profiles, balance equations, and assumptions are combined to give a system of ordinary differential equations as follows:

$$\left|\begin{array}{cccc} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{array}\right| \left|\begin{array}{c} \dfrac{dc_c}{ds} \\ \dfrac{d}{ds}(\sigma_{y'}\sigma_{z'}) \\ \dfrac{d\theta}{ds} \\ \dfrac{du_c}{ds} \end{array}\right| = \left|\begin{array}{c} B_1 \\ B_2 \\ B_3 \\ B_4 \end{array}\right|$$

where

(Contaminant Mass Balance)

$$A_{11} = (k_1 u_a \cos\theta + k_2 u_c)\,\sigma_{y'}\sigma_{z'}$$

$$A_{12} = (k_1 u_a \cos\theta + k_2 u_c)\, c_c$$

$$A_{13} = -k_1 u_a \sin\theta\, c_c \sigma_{y'} \sigma_{z'}$$

$$A_{14} = k_2 c_c \sigma_{y'} \sigma_{z'}$$

$$B_1 = 0$$

(Total Mass Balance)

$$A_{21} = 0$$

$$A_{22} = \rho_a (k_1 u_a \cos\theta + k_2 u_c)$$

$$A_{23} = -\rho_a k_3 u_a \sin\theta\, \sigma_{y'} \sigma_{z'}$$

$$A_{24} = \rho_a k_4 \sigma_{y'} \sigma_{z'}$$

$$B_2 = E_1 + E_2 + E_3$$

(z-Direction Momentum Balance)

$$A_{31} = (k_1 u_a^2 \cos^2\theta \sin\theta + 2k_2 u_a u_c \cos\theta \sin\theta + k_6 u_c^2 \sin\theta)\, \gamma \sigma_{y'} \alpha_{z'}$$

$$A_{32} = \rho_a (k_3 u_a^2 \cos^2\theta \sin\theta + 2k_4 u_a u_c \cos\theta \sin\theta + k_5 u_c^2 \sin\theta) + (k_1 u_a^2 \cos^2\theta \sin\theta + 2k_2 u_a u_c \cos\theta \sin\theta + k_6 u_c^2 \sin\theta)\, \gamma c_c$$

$$
\begin{aligned}
A_{33} = \rho_a(&-2k_3u_a^2 \cos\theta \sin^2\theta + k_3u_a^2 \cos^3\theta \\
&-2k_4u_au_c \sin^2\theta + 2k_4u_au_c \cos^2\theta \\
&+ k_5u_c^2 \cos\theta)\ \sigma_{y'}\sigma_{z'} \\
&+ (-2k_1u_a^2 \cos\theta \sin^2\theta + k_1u_a^2 \cos^3\theta \\
&- 2k_2u_au_c \sin^2\theta + 2k_2u_au_c \cos^2\theta \\
&+ k_6u_c^2 \cos\theta)\ \gamma c_c\sigma_{y'}\sigma_{z'}
\end{aligned}
$$

$$
\begin{aligned}
A_{34} = \rho_a&(2k_4u_a \cos\theta \sin\theta + 2k_5u_c \sin\theta)\ \sigma_{y'}\sigma_{z'} \\
&+ (2k_2u_a \cos\theta \sin\theta + 2k_6u_c \sin\theta)\ c_c\sigma_{y'}\sigma_{z'}
\end{aligned}
$$

$$B_3 = gk_1\gamma c_c\sigma_{y'}\sigma_{z'}$$

(x-Direction Momentum Balance)

$$A_{41} = (k_1u_a^2 \cos^3\theta + 2k_2u_au_c \cos^2\theta + k_6u_c^2 \cos\theta)\ \gamma\sigma_{y'}\sigma_{z'}$$

$$
\begin{aligned}
A_{42} = \rho_a&(k_3u_a^2\cos^3\theta + 2k_4u_au_c \cos^2\theta + k_5u_c^2 \cos\theta) \\
&+ (k_1u_a^2 \cos^3\theta + 2k_2u_au_c \cos^2\theta + k_6u_c^2 \cos\theta)\ \gamma c_c
\end{aligned}
$$

$$
\begin{aligned}
A_{43} = \rho_a(&-{}_3u_a^2 \cos^2\theta \sin\theta - 4k_4u_au_c \cos\theta \sin\theta \\
&- k_5u_c^2 \sin\theta)\ \sigma_{y'}\sigma_{z'} \\
&+ (-3k_1u_a^2 \cos^2\theta \sin\theta - 4k_2u_au_c \cos\theta \sin\theta \\
&- k_6u_c^2 \sin\theta)\ \gamma c_c\sigma_{y'}\sigma_{z'}
\end{aligned}
$$

$$
\begin{aligned}
A_{44} = \rho_a&(2k_4u_a \cos^2\theta + 2k_5u_c \cos\theta)\ \sigma_{y'}\sigma_{z'} \\
&+ (2k_2u_a \cos^2\theta + 2k_6u_c \cos\theta)\ \gamma c_c\sigma_{y'}\sigma_{z'}
\end{aligned}
$$

$$B_4 = u_a(E_1 + E_2 + E_3)$$

and

$$k_1 = 2\pi \left[\operatorname{erf} \left[\frac{\delta}{2} \left[\frac{\pi}{2} \right]^{\frac{1}{2}} \right] \right]^2$$

$$k_2 = \left[\frac{2\pi}{1 + \lambda^2} \right] \left[\operatorname{erf} \left[\frac{\delta}{2} \left[\frac{\pi(1 + \lambda^2)}{2} \right]^{\frac{1}{2}} \right] \right]^2$$

$$k_3 = \pi\delta^2$$

$$k_4 = \frac{2\pi}{\lambda^2} \left[\operatorname{erf} \left[\frac{\delta\lambda}{2} \left[\frac{\pi}{2} \right]^{\frac{1}{2}} \right] \right]^2$$

$$k_5 = \frac{\pi}{\lambda^2} \left[\operatorname{erf} \left[\frac{\delta\lambda \sqrt{\pi}}{2} \right] \right]^2$$

$$k_6 = \left[\frac{2\pi}{1 + 2\lambda^2} \right] \left[\operatorname{erf} \left[\frac{\delta}{2} \left[\frac{\pi(1 + 2\lambda^2)}{2} \right]^{\frac{1}{2}} \right] \right]^2$$

After c_c is determined from the system of equations above, the contribution from the reflected image is approximated by

$$c_c \exp \left\{ -\frac{1}{2} \left[\frac{2\, z_j}{\sigma_{z'}} \right]^2 \right\} \qquad \text{(III.31)}$$

so that the reported concentration is

$$c_c'' = c_c \left\{ 1 + \exp \left\{ -\frac{1}{2} \left[\frac{2\, z_j}{\sigma_{z'}} \right]^2 \right\} \right\} \qquad \text{(III.32)}$$

The reported mole fraction and density are calculated based on c_c''. Although the method of images is a common approach for accounting for ground interactions with the plume, application of the method of images to this problem may cause c_c'' to increase with downwind distance. Note that at the centerline touchdown, $c_c'' = 2c_c$.

For Ooms model predictions, DEGADIS is used to model the subsequent dispersion when the jet-plume reaches ground level ($z_j = 0$). The ground-level source input to DEGADIS is modeled as a circular area source with radius $\delta\sigma_{y'}$ and concentration equal to the centerline value from the Ooms model. Note that the downwind distance reported in DEGADIS includes the distance to touchdown calculated in the Ooms model. A description of the DEGADIS model is included in Section V.

IV. EVALUATION OF THE JET-PLUME MODEL

A series of simulations were made with the jet-plume model to:

- compare the model predictions with wind tunnel dense gas jet trajectory and dilution data
- characterize the sensitivity of the model to the specification of entrainment coefficients α_1 and α_2.

Table IV.1 shows the "typical" jet releases simulated.

Table IV.1. Specification of Gas Release Rates for Modeling.

Jet Diameter m	Jet Velocity m/s	Gas Release Rate kg/s
0.05 (~2 in)	30.6 (~100 ft/s)	0.24
0.2 (~8 in)	91.7 (~300 ft/s)	11.52
0.5 (~20 in)	213.9 (~700 ft/s)	168.0

Comparison with Wind Tunnel Test Data

Hoot, Meroney, and Peterka (1973) reported plume rise, downwind distance to plume centerline touchdown, and dilution at touchdown for wind tunnel jet releases of Freon-12/air mixtures. The ranges of experimental variables studied were:

gas specific gravity (air = 1)	1.1 - 4.6
gas exit diameter, cm	0.32 and 0.64
gas exit height, cm	7.6 and 15.2
gas exit velocity/wind velocity ratio	2.5 - 25
wind (tunnel) velocity, m/s	0.23 and 0.46

Correlations (of the wind tunnel data) were presented for plume rise, distance to plume touchdown, and plume centerline concentration at (centerline) ground contact, in a laminar crosswind:

Plume Rise

$$H_r = 1.32\ D\ [(V/u_a)(\rho_e/\rho_a)]^{1/3} Fr_r^{2/3} \tag{IV.1}$$

Downwind Distance to Maximum Rise

$$\overline{X} = (D\ u_a/V)\ Fr_r^2 \tag{IV.2}$$

Downwind Distance to Centerline Ground Contact

$$X_c = 0.56\ D\ \{(H_r/D)^3[(2 + H_s/H_r)^3 - 1]u_a/V\}^{1/2}Fr_h + \overline{X} \qquad \text{(IV.3)}$$

Centerline Concentration at Ground Contact

$$\chi_c = \frac{6.0\ Q}{u_a(2H_r + H_s)^2} \qquad \text{(IV.4)}$$

where D = initial jet diameter (m)

Fr = "release" Froude number, $V/[g\ D\ (\rho_e-\rho_a)/\rho_\rho]^{1/2}$

Fr_h = "horizontal" Froude number, $u_a/[g\ D\ (\rho_e-\rho_a)/\rho_a]^{1/2}$

H_r = maximum height of plume rise above stack (m)

H_s = exit height (m)

Q = steady-state jet rate (kg/s)

u_a = wind velocity (m/s)

V = initial jet velocity (m/s)

X_c = downwind distance to ground contact (centerline) (m)

$\overline{X}$ = downwind distance to maximum rise (m)

ρ_a = ambient air density (kg/m^3)

ρ_e = initial jet density (kg/m^3)

χ_c = jet centerline concentration at centerline touchdown (kg/m^3)

Note that Equation (IV.4) is based on analysis by Briggs presented in Guinnup (1989).

Table IV.2 compares the jet-plume model predictions for plume rise, downwind distance to ground contact, and concentration at ground contact with the results of Hoot et al.'s wind-tunnel data correlations for the "low diameter/low velocity", "typical diameter/typical velocity", and "high diameter/high velocity" cases (Table IV.1) in 3 and 6 m/s winds. Based on a scaled release height of 10 m, a surface roughness of 2.2 mm was assumed with the 3 and 6 m/s velocities because the jet-plume model implementation assumes a logarithmic velocity profile. This assumption causes overprediction of the wind tunnel velocities by greater than 20% for scaled heights greater than ~40 m. Since the wind tunnel correlations were for jets in a laminar crosswind, the simulations shown in Table IV.2 were made with the entrainment due to atmospheric turbulence (E_3) set to zero.

Table IV.2. Comparison of Jet-Plume Model Prediction with Hoot et al.'s Wind Tunnel Data Correlation

Gas Jet Density: 4.0 kg/m^3 Gas Jet Elevation: 10 m

(Hoot et al. correlation / Jet-Plume model prediction)

Maximum Rise, m (3, 6 m/s)		Distance to Ground Contact, m (3, 6 m/s)		Centerline Concentration at Ground Contact kg/m^3 x 10^3 (3, 6 m/s)	
Low Diameter (0.05 m) - Low Velocity (30.6 m/s)					
3.0/2.6	2.4/1.7	150/160	375/370	1.9/4.2	1.1/2.6
Typical Diameter (0.2 m) - Typical Velocity (91.7 m/s)					
22.6/20.9	17.9/15.2	165/170	350/320	7.6/17.4	5.4/14.3
High Diameter (0.5 m) - High Velocity (213.9 m/s)					
97.0/87.4	76.9/66.5	320/380	650/680	8.1/18.3	6.3/15.5

The model predictions of maximum rise and distance to centerline ground contact are in good agreement with the wind tunnel correlations. The agreement between model predictions of distance to centerline ground contact and the wind tunnel correlations tends to degrade as the maximum rise increases because the logarithmic velocity profile assumed in the model overpredicts the (laminar) wind tunnel velocity. The model prediction of the maximum concentration at centerline ground contact differs by a factor of about two from the observed concentration. Note that if interaction with the ground were not included in the calculations, the predicted model concentrations would be reduced by exactly a factor of 2. Therefore, the agreement between model predictions of the maximum concentration at centerline ground contact and the wind tunnel correlations is considered good.

Sensitivity to Entrainment Coefficient Specification

The model sensitivity to the coefficients α_1 and α_2 was determined by systematic variation around the values given above. All simulations were of denser-than-air gas jets (initial density 4.0 kg/m^3) exiting vertically upward at 10 meters elevation. The predictions assumed a logarithmic velocity profile with 3 and 6 m/s velocities at 10 m elevation, a surface roughness of 2.2 mm, and D stability. Table IV.3 shows the effect of individually varying the entrainment coefficients α_1 and α_2 by factors of two below and above the values given above, for the "low diameter/low velocity", "typical diameter/typical velocity", and "high diameter/high velocity" cases (Table IV.1) in 3 and 6 m/s winds.

The predictions of maximum rise and distance to centerline ground contact are relatively insensitive to factor-of-four variations in α_1 and α_2 over the range of release conditions in Table IV.1. The predictions of maximum concentration at centerline ground contact are more sensitive to variations in α_1 and α_2; factor-of-four variations in α_1 and α_2 resulted in concentration changes of less than a factor of ~2 when concentrations were compared at the same distances.

Table IV.3. Sensitivity of Jet-Plume Model Prediction to Variation of Entrainment Coefficients α_1 and α_2

Gas Jet Density: 4.0 kg/m^3 **Gas Jet Elevation: 10 m**

Entrainment Coefficients		Maximum Rise, m (3, 6 m/s)		Distance to Ground Contact, m (3, 6 m/s)		Centerline Concentration at Ground Contact kg/m^3 x 10^3 (3, 6 m/s)	
α_1	α_2						
Low Diameter (0.05 m) - Low Velocity (30.6 m/s)							
0.014	0.37	2.9	1.6	740*	420*	--	--
0.028	0.37	2.5	1.5	750*	420*	--	--
0.056	0.37	2.1	1.4	750*	420*	--	--
0.028	0.185	2.9	1.7	750*	420*	--	--
0.028	0.37	2.5	1.5	750*	420*	--	--
0.028	0.74	2.1	1.2	740*	420*	--	--
Typical Diameter (0.2 m) - Typical Velocity (91.7 m/s)							
0.014	0.37	25.7	16.9	300	1770	4.7	0.22
0.028	0.37	20.6	14.4	240	1440	6.2	0.31
0.056	0.37	16.4	12.2	210	1190	7.7	0.42
0.028	0.185	23.0	16.4	220	1530	9.1	0.30
0.028	0.37	20.6	14.4	240	1440	6.2	0.31
0.028	0.74	17.3	11.9	280	1400	3.6	0.30
High Diameter (0.5 m) - High Velocity (213.9 m/s)							
0.014	0.37	112	79.4	510	1280	10.2	3.2
0.028	0.37	86.6	64.7	420	1030	13.1	4.4
0.056	0.37	66.9	52.6	350	860	15.4	5.7
0.028	0.185	95.0	73.4	380	1010	21.4	5.6
0.028	0.37	86.6	64.7	420	1030	13.1	4.4
0.028	0.74	74.7	53.7	480	1100	6.8	3.0

*The plume remained aloft. The reported distance is the estimated distance to 10 ppm.

V. DESCRIPTION OF THE DEGADIS DENSE GAS DISPERSION MODEL

DEGADIS (DEnse GAs DISpersion) model development was sponsored by the U.S. Coast Guard and the Gas Research Institute (Havens and Spicer, 1985). DEGADIS is an adaptation of the Shell HEGADAS model (Colenbrander, 1980 and Colenbrander and Puttock, 1983) and incorporates some techniques proposed by van Ulden (1983). It is designed to model the atmospheric dispersion of ground-level, area-source dense gas (or aerosol) clouds released with zero (initial) momentum into an atmospheric boundary layer flow over flat, level terrain.

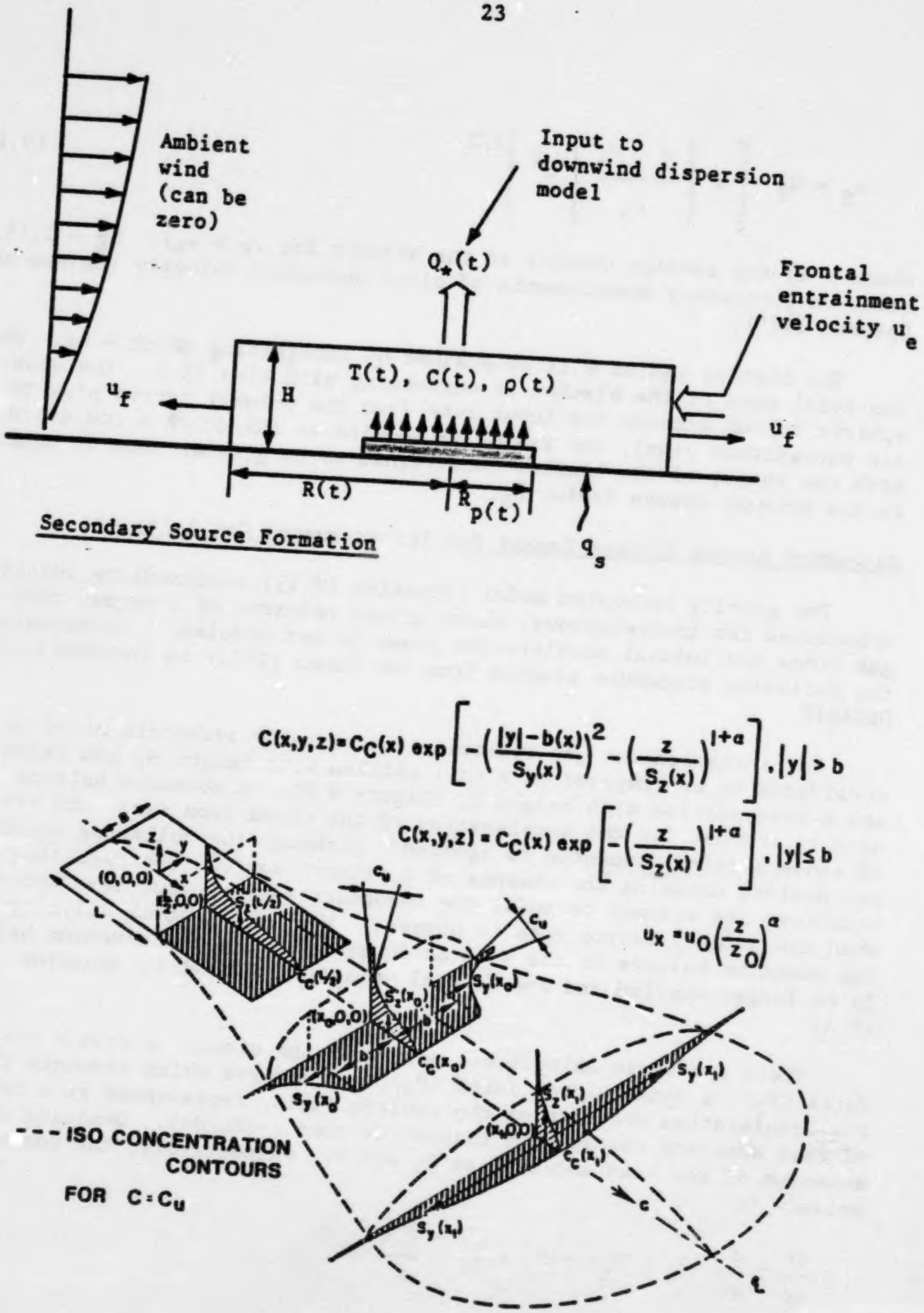
If a gas release rate does not exceed the potential atmospheric takeup rate, the gas is taken up directly by the atmospheric flow and dispersed downwind. If a gas release rate exceeds the potential atmospheric takeup rate, a denser-than-air "secondary source" blanket is formed over the primary source, and this near-field, buoyancy-dominated regime is modeled using a lumped parameter model (spatially averaged properties) which incorporates air entrainment at the gravity-spreading front using a frontal entrainment velocity (Figure V.1). Downwind dispersion is modeled with a power law concentration distribution in the vertical direction and a modified Gaussian profile in the horizontal direction with a power law specification for the wind profile. The downwind dispersion models the ensemble average of the gas concentrations.

Secondary Source Blanket Formation

A denser-than-air gas "secondary source" cloud or blanket may be formed over an evaporating liquid pool (or otherwise specified ground-level emission source) or by the instantaneous release of a volume of gas. A lumped parameter model of such a blanket is illustrated in Figure V.1. The blanket is represented as a cylindrical gas volume which spreads laterally as a density-driven flow with entrainment from the top of the source blanket by wind shear and air entrainment into the advancing front edge. The blanket spreads laterally until the atmospheric takeup rate from the top is matched by the air entrainment rate from the side and, if applicable, by the rate of gas addition from under the blanket. The blanket center is assumed stationary over the source.

Secondary Source Blanket Extent for Ground-Level Emission Sources

The downwind emission rate from the blanket is equal to the potential atmospheric takeup rate, Q_{*max}. For $E(t)/\pi R_p^2(t) > Q_{*max}$, a blanket is formed over the primary source. The blanket frontal (spreading) velocity is modeled as


Ambient wind (can be zero)
Input to downwind dispersion model
$Q_*(t)$
Frontal entrainment velocity u_e
$T(t)$, $C(t)$, $\rho(t)$
u_f
H
u_f
$R(t)$
$R_p(t)$
q_s
Secondary Source Formation
(0,0,0)
($\frac{L}{2}$,0,0)
$S_z(L/2)$
$C_c(L/2)$
L
c_u
c_u
$S_z(x_0)$
$S_y(x_0)$
$(x_0,0,0)$
$C_c(x_0)$
$S_y(x_0)$
$S_z(x_1)$
$S_y(x_1)$
$(x_1,0,0)$
$C_c(x_1)$
$S_y(x_1)$
c
℄
--- ISO CONCENTRATION CONTOURS FOR $C = C_u$

$$C(x,y,z) = C_C(x) \exp\left[-\left(\frac{|y|-b(x)}{S_y(x)}\right)^2 - \left(\frac{z}{S_z(x)}\right)^{1+\alpha}\right], \; |y| > b$$

$$C(x,y,z) = C_C(x) \exp\left[-\left(\frac{z}{S_z(x)}\right)^{1+\alpha}\right], \; |y| \le b$$

$$u_x = u_O\left(\frac{z}{z_O}\right)^{\alpha}$$

Figure V.1. Schematic Diagram of DEGADIS Dense Gas Dispersion Model.

$$u_f = C_E \left[g \left(\frac{\rho - \rho_a}{\rho_a} \right) H^2 \right]^{1/2} \qquad \text{(V.1)}$$

where ρ is the average density of the blanket for $(\rho > \rho_a)$. $C_E = 1.15$, based on laboratory measurements of cloud spreading velocity (Havens and Spicer, 1985).

The blanket radius R is determined by integrating $dR/dt = u_f$. When the total mass of the blanket is decreasing with time (i.e., the atmospheric takeup exceeds the input rate from the primary source plus the air entrainment rate), the radius is modeled as $(dR/dt)/R = (dH/dt)/H$, with the radius of the blanket constrained to be greater than or equal to the primary source radius R_p.

Secondary Source Blanket Extent for Instantaneous Gas Releases

The gravity intrusion model (Equation (V.1)) overpredicts initial velocities for instantaneous, above-ground releases of a denser-than-air gas since the initial acceleration phase is not modeled. Consequently, the following procedure adapted from van Ulden (1983) is incorporated in DEGADIS.

For instantaneous gas releases, the radially symmetric cloud is considered to be composed of a tail section with height H_t and radius R_h and a head section with height H_h (Figure V.2). A momentum balance is used to account for the acceleration of the cloud from rest; the effect of ambient (wind) momentum is ignored. Although the following equations are derived assuming the absence of a primary source, the resulting equations are assumed to model the secondary source cloud development when the primary source rate is nonzero. When the frontal velocity from the momentum balance is the same as Equation (V.1), the momentum balance is no longer applied and the frontal velocity is given by Equation (V.1).

There are three main forces acting on the cloud: a static pressure force (F_p), a dynamic drag force (F_d), and a force which accounts for the acceleration reaction of the ambient fluid, represented as a rate of virtual momentum change with respect to time ($-dP_v/dt$). Denoting the momentum of the head and tail as P_h and P_t, respectively, the momentum balance is

$$\frac{dP}{dt} = \frac{d}{dt}(P_h + P_t) = F_p + F_d - \frac{dP_v}{dt} \qquad \text{(V.2)}$$

or

$$\frac{d}{dt}(P_h + P_t + P_v) = F_p + F_d \qquad \text{(V.3)}$$

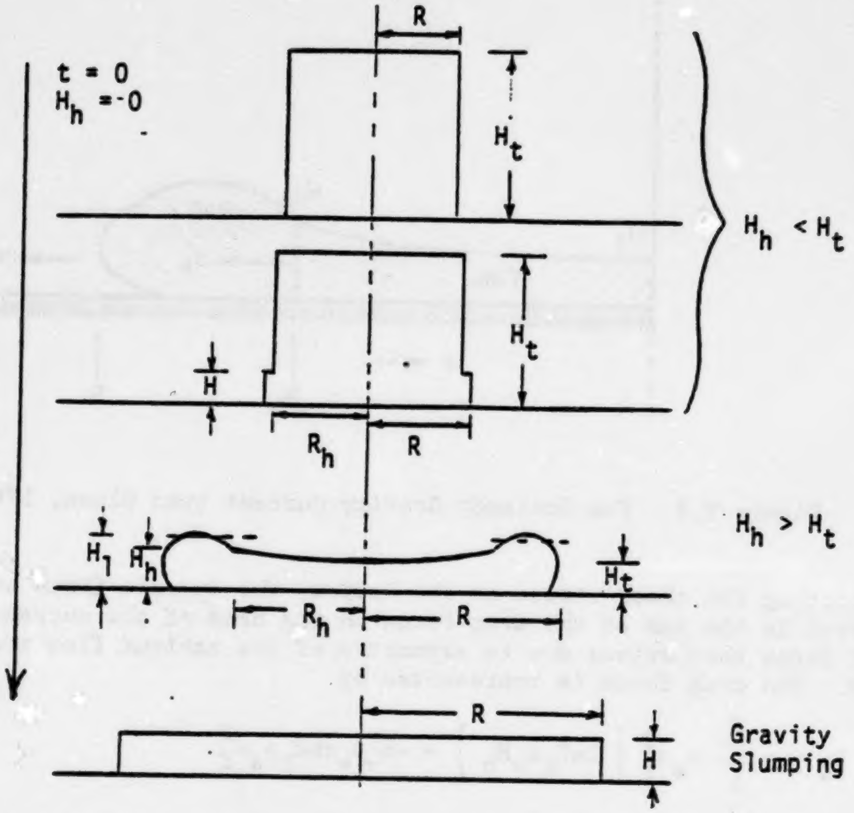


Figure V.2. Schematic Diagram of a Radially Spreading Cloud.

The terms in the momentum balance are evaluated differently for the time periods before a gravity current head has developed ($H_h < H_t$) and after the head has developed but the cloud is still accelerating (Figure V.2). Model equations describing the time period after the gravity current head forms ($H_h \geq H_t$) are derived first. The model equations describing the earlier time period ($H_h < H_t$) use simplifications of the equations applicable for $H_h \geq H_t$.

Unsteady Gravity Current ($H_h \leq H_t$)

When $H_h \geq H_t$ (Figures V.2, V.3), the frontal velocity is determined from the momentum balance (Equation (V.2)) as follows.

The static pressure force, obtained by integrating the static pressure over the boundary of the current, is

$$F_p = \left[\frac{1}{2} g\Delta\rho H_t \right] \left[2\pi R H_t \right] = \pi g \Delta\rho R H_t^2 \qquad (V.4)$$

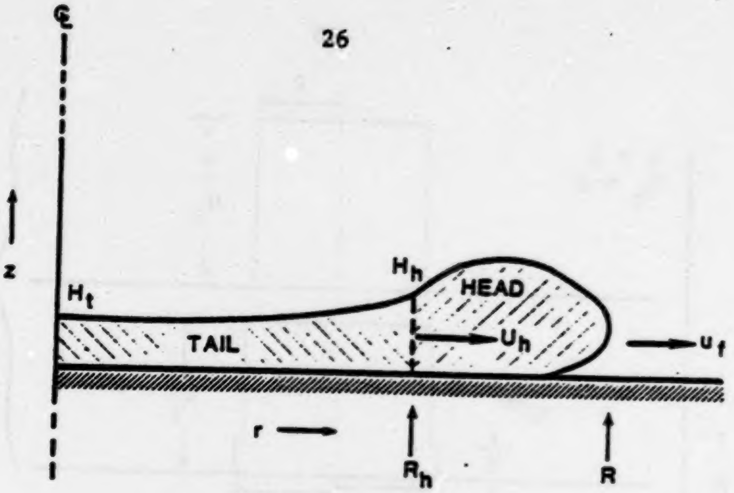


Figure V.3. The Unsteady Gravity Current (van Ulden, 1983).

Neglecting the shear stress at the bottom, the dynamic force on the current is the sum of the drag force on the head of the current and the lift force that arises due to asymmetry of the ambient flow around the head. The drag force is represented by

$$F_D = -\frac{d_v}{2}\rho_a u_f^2\left(2\pi R_h a_v H_h\right) = -a_v d_v \pi R H_h \rho_a u_f^2 \qquad \text{(V.5)}$$

where d_v is an effective drag coefficient and the constant a_v is an empirical ratio of the average head depth H_1 to H_h ($a_v = H_1/H_h$).

The horizontal acceleration reaction ($-dP_v/dt$) is approximated by the reaction to an accelerating elliptical cylinder with an aspect ratio H/R (Batchelor, 1967):

$$-\left(\frac{dP_v}{dt}\right) = -\frac{d}{dt}\left(k_1\rho_a\pi RH^2u_f\right) \qquad \text{(V.6)}$$

and the vertical acceleration reaction is represented as

$$-\left(\frac{dP_v}{dt}\right)_z = -\frac{d}{dt}\left(k_2\rho_a\pi RH^2u_f\right) \qquad \text{(V.7)}$$

where k_1 and k_2 are coefficients of order one. Using a single constant, Equations (V.6) and (V.7) give

$$-\frac{dP_v}{dt} = -e_v\pi\rho_a\frac{d(RH^2u_f)}{dt} \qquad \text{(V.8)}$$

Using Equations (V.4), (V.5), and (V.8), the momentum balance (Equation (V.2)) becomes

$$\frac{dP}{dt} = \pi g \Delta\rho R H_t^2 - a_v d_v \pi \rho_a R H_h u_f^2 - e_v \pi \rho_a \frac{d(RH^2 u_f)}{dt} . \qquad (V.9)$$

Following van Ulden (1979, 1983), it is assumed that the potential energy decrease due to slumping of the cloud is offset by the production of kinetic energy, which through the action of shear is partly transformed to turbulent kinetic energy. Part of the turbulent kinetic energy is transformed back into potential energy due to entrainment of air by the cloud. This "buoyant destruction" of kinetic energy is assumed to be proportional to the rate of production of turbulent kinetic energy, and following Simpson and Britter (1979) it is assumed that the turbulent kinetic energy production rate scales as $\pi\rho_a HRu_f^3$. Then,

$$\frac{1}{2} g \Delta\rho H \frac{dV}{dt} = \epsilon \pi \rho_a H R u_f^3 \qquad (V.10)$$

which can be written

$$\frac{dV}{dt} = \frac{\epsilon(2\pi RH)u_f}{\left[\dfrac{g\Delta\rho H}{\rho_a u_f^2}\right]} = \frac{\epsilon(2\pi RH)u_f}{Ri_f} \qquad (V.11)$$

where ϵ is an empirical coefficient. Noting that dV/dt represents the air entrainment rate,

$$\frac{\dot{M}_a}{\rho_a} = \epsilon(2\pi RH)u_f \Bigg/ \left[\frac{g\Delta\rho H}{\rho_a u_f^2}\right] \qquad (V.12)$$

where $\dot{M}_a$ represents the air entrainment mass rate.

The volume integral

$$V = 2\pi \int_0^R h(r,t) r dr \qquad (V.13)$$

where h(r,t) is to be expressed in terms of H_h and H_t, and the momentum integral

$$P = 2\pi \int_0^R \rho u(r,t) h(r,t) r dr = P_t + P_h \qquad (V.14)$$

are then approximated with separate analyses of the head and tail of the current.

In the tail of the current, the shallow layer approximation is applied. It is assumed that the shape of the current is quasi-stationary in time, and the layer-averaged density difference is assumed horizontally uniform. It follows that the volume and momentum of the tail are given by

$$V_t = \pi R_h^2 \left[H_t + H_h \right] / 2. \tag{V.15}$$

$$P_t = \frac{2}{5} \rho \left[\frac{2}{3} H_t + H_h \right] \pi R_h^3 \frac{u_f}{R} \tag{V.16}$$

A momentum balance for the head region, Figure V.4, assuming quasi-steady state, indicates that the static and dynamic pressure forces on the head should be balanced by the net flux of momentum due to flow into and out of the head. The static pressure and drag are, respectively

$$F_P = \left[\frac{1}{2} g\Delta\rho H_h \right] \left[2\pi R_h H_h \right] = \pi g\Delta\rho R_h H_h^2 \tag{V.17}$$

and

$$F_D = - d_v \left[\frac{1}{2} \rho_a u_f^2 \right] \left[2\pi R_h (a_v H_h) \right]$$

$$= - a_v d_v \rho_a u_f^2 \pi R_h H_h \tag{V.18}$$

Near the surface, the inward flow (u_4 in Figure V.4) carries momentum into the head, while the return flow (u_3 in Figure V.4) carries momentum out of the head. Assuming $u_3 = u_4$, $H_4 = 1/2\ H_h$, and $u_4 = \delta_v u_f$, the momentum flux into the head is approximately

$$Q_h = \delta_v^2 \rho_a u_f^2 \left[2\pi R_h H_h \right] \tag{V.19}$$

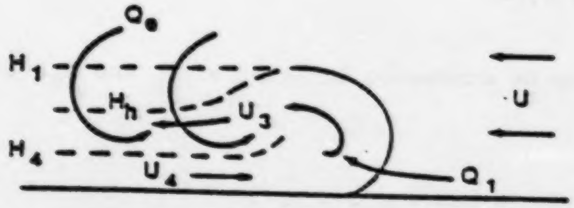


Figure V.4. The Head of a Steady Gravity Current (Simpson and Britter, 1979; van Ulden, 1983).

Upon rearranging, the momentum balance on the head gives

$$\frac{\rho_a u_f^2}{g\Delta\rho H_h} = 1./\left[a_v d_v - 2\delta_v^2 \right] = C_E^2 \tag{V.20}$$

when $\delta_v = 0.2$ and $d_v = 0.64$; Equation (V.20) then specifies the head velocity boundary condition. The volume of the head is determined by assuming that the head length scales with H_1. It follows that

$$R - R_h = b_v H_1 \tag{V.21}$$

where b_v is an empirical constant, and the volume of the head becomes

$$V_h = \pi a_v^2 b_v \; (R + R_h) \; H_h^2 \tag{V.22}$$

If the layer-averaged velocity is assumed to increase linearly with r, it follows that

$$u_h = u_f \left[\frac{R_h}{r} \right] \tag{V.23}$$

and

$$P_h = \frac{2\pi}{3} \rho a_v \frac{u_f H_h}{R} \left[R^3 - R_h^3 \right] \tag{V.24}$$

Along with the definition of u_f,

$$\frac{dR}{dt} = u_f, \tag{V.25}$$

Equations (V.9), (V.11), (V.20), (V.21), (V.23), and (V.25) are solved to determine ρ, H_t, H_h, V, P_h, and P_t when $H_h > H_t$.

The constants a_v, b_v, d_v, e_v, and ϵ are assigned values 1.3, 1.2, 0.64, 20., and 0.59, respectively, based on analysis of the still-air denser-than-air gas release experiments of Havens and Spicer (1985).

Initial Gravity Current Development ($H_h < H_t$)

In order to model the initial cloud shape, the tail and head height are considered constant with respect to radius. The momentum balance on the cloud is then given by

$$\frac{d}{dt} \left[P_h + P_t \right] = \pi g \Delta\rho \left[R_h \; H_t^2 + a_v b_v \; H_h^3 \right]$$

$$- \pi a_v d_v \rho_a R H_h u_f^2 - \frac{dP_v}{dt} \tag{V.26}$$

where the first term on the right-hand side represents the static pressure force on the head and the second term represents the drag force on the bottom surface of the cloud. The third force is the acceleration reaction by the ambient fluid, represented by Equation (V.8).

The dimensions of the head are again given by

$$R_h = R - a_v b_v H_h \tag{V.27}$$

and

$$H_h = \left(\frac{u_f}{C_E} \right)^2 \bigg/ [g\Delta\rho/\rho_a] \tag{V.28}$$

When the height of the tail H_t is assumed uniform with respect to radius, it follows that

$$H_t = \left[\frac{M}{\rho} - \pi a_v^2 b_v \,(R + R_h)\, H_h^2 \right] \bigg/ (\pi R_h^2) \tag{V.29}$$

where M is the total mass of the cloud. The momentum of the head P_h and tail P_t are then

$$P_h = \frac{2}{3} \pi a_v \frac{\rho H_h \,(R^3 - R_h^3)}{R} u_f \tag{V.30}$$

and

$$P_t = \frac{2}{3} \pi \frac{\rho H_t R_h^3}{R} u_f \tag{V.31}$$

Equations (V.26) through (V.31) determine the momentum of the blanket as a function of time, and thus the frontal velocity u_f. The cloud accelerates from rest because $H_h = 0$ initially.

Material and Energy Balances

The balance on the total mass of gas in the source blanket ($M = \pi R^2 H\rho$) is

$$\frac{dM}{dt} = \frac{d}{dt} \left[\pi R^2 H\rho \right] = \frac{E(t)}{w_{c,p}(t)} + \dot{M}_a + \dot{M}_{w,s} - \left[\frac{Q_{*max}}{w_c} \right] \left[\pi R^2 \right] \tag{V.32}$$

where E(t) is the contaminant evolution rate from the primary (liquid) source and $w_{c,p}(t)$ is the contaminant mass fraction in the primary source. For spills onto water, the water entrainment term ($\dot{M}_{w,s}$) is included in the source blanket description and is calculated from

Equation (V.46), and the (humid) air entrainment rate (Equation (V.12)) is

$$\dot{M}_a = 2\pi RH(\epsilon u_f)\rho_a \Big/ \left[g\Delta\rho H/(\rho_a u_f^2) \right] \tag{V.33}$$

The balance on the mass of contaminant in the source blanket $(M_c = w_c \pi R^2 H\rho)$ is

$$\frac{dM_c}{dt} = \frac{d}{dt}\left[w_c \pi R^2 H\rho \right] = E(t) - Q_{*max}\ (\pi R^2) \tag{V.34}$$

and the mass balance on the air in the source blanket $(M_a = w_a \pi R^2 H\rho)$ is

$$\frac{dM_a}{dt} = \frac{d}{dt}\left[w_a \pi R^2 H\rho \right] = \frac{E(t)}{w_{c,p}(t)} \left(\frac{1 - w_{c,p}(t)}{1 + H_a} \right) + \frac{\dot{M}_a}{1 + H_a} - \left(\frac{Q_{*max}}{w_c} \right) w_a\ (\pi R^2) \tag{V.35}$$

where the ambient humidity is H_a and the mass fraction of contaminant and air are $w_c = M_c/M$ and $w_a = M_a/M$, respectively. Note that any dilution of the primary source with air is assumed to be at ambient humidity.

The energy balance on the source blanket $(h\pi R^2 H\rho)$ gives

$$\frac{d}{dt}\left[h\pi R^2 H\rho \right] = \frac{h_p\ E(t)}{w_{c,p}(t)} + h_a\dot{M}_a + h_w\dot{M}_{w,s} - h\left(\frac{Q_{*max}}{w_c} \right)(\pi R^2) + \dot{Q}_s \tag{V.36}$$

where h_p is the enthalpy of the primary source gas, h_a is the enthalpy of the ambient humid air, and h_w is the enthalpy of any water vapor entrained by the blanket (if on water). There are three alternate submodels included for heat transfer $(\dot{Q}_s)$ from the surface to the cloud.

The simplest method for calculating the heat transfer between the substrate and the gas cloud is to specify a constant heat transfer coefficient for the heat transfer relation

$$\dot{Q}_s = q_s\left[\pi\left(R^2 - R_p^2 \right) \right] = h_0\Delta T\left[\pi\left(R^2 - R_p^2 \right) \right] \tag{V.37}$$

where $\mathring{Q}_s$ is the rate of heat transfer to the cloud, q_s is the heat flux, and ΔT is the temperature difference between the cloud and the surface. For the calculation of heat transfer over the source, the temperature difference is based on the average temperature of the blanket.

In the evaluation of the Burro and Coyote series of experiments, Koopman et al. (1981) proposed the following empirical heat transfer coefficient relationship for heat transfer between a cold LNG vapor cloud and the ground

$$h_0 = V_H \rho C_p \tag{V.38}$$

where the value of V_H was estimated to be 0.0125 m/s. (The constant V_H can be specified in this model.)

From the heat transfer coefficient descriptions for heat transfer from a flat plate, the following relationships can be applied. For natural convection, the heat transfer coefficient is estimated using the Nusselt (Nu), Grashoff (Gr), and Schmidt (Sc) numbers (McAdams, 1954) from

$$Nu = 0.14\ (Gr\ Sc)^{1/3} \tag{V.39}$$

or

$$h_n = 0.14 \left(\frac{g \rho^2 C_p^3 \mu}{T\ Pr^2} \Delta T \right)^{1/3} \tag{V.40}$$

where h_n is the natural convection heat transfer coefficient and Pr is the Prandtl number. In order to simplify the calculations, the parameter group

$$\left[Pr^{-2} \left(C_p MW \right)^3 \left(\frac{\mu}{MW\ T} \right) \right]^{1/3} \tag{V.41}$$

is estimated to be 60 in mks units (actual values are 47.25, 58.5, and 73.4 for air, methane, and propane, respectively). Equation (V.40) becomes

$$h_n = 18 \left[\left(\frac{\rho}{MW} \right)^2 \Delta T \right]^{1/3} \tag{V.42}$$

where the density ρ, molecular weight MW, and temperature difference ΔT are based on the average composition of the gas blanket.

For forced convection, the Colburn analogy (Treybal, 1980) is applied to a flat plate using the Stanton number for heat transfer St_H and the Prandtl number as

$$St_H \, Pr^{2/3} = \frac{c_f}{2} = \left[\frac{u_*}{\overline{u}} \right]^2 \tag{V.43}$$

or

$$h_f = (\overline{u}\rho C_p) \, Pr^{-2/3} \left[\frac{u_*}{\overline{u}} \right]^2 \tag{V.44}$$

where h_f is the forced convection heat transfer coefficient. If the velocity is evaluated at $z = H/2$ and Pr is specified 0.741,

$$h_f = \left[1.22 \, \frac{u_*^2}{u_0} \left[\frac{2z_0}{H} \right]^{\alpha} \right] \rho C_p \tag{V.45}$$

If $H/2 < z_R$, then the velocity is evaluated at $z = z_R$.

The overall heat transfer coefficient is then the maximum of the forced and natural convection coefficients, i.e. $h_0 = \max(h_f, h_n)$. The heat flux and transfer rate are then estimated by Equation (V.37).

If the gas blanket is formed over water, water may be transferred from the water surface to the blanket. The rate of mass transfer of water is modeled as

$$\dot{M}_{w,s} = \frac{F_0}{p} \left[p_{w,s}^* - p_{w,c} \right] \left[\pi \left[R^2 - R_p^2 \right] \right] \tag{V.46}$$

where F_0 is an overall mass transfer coefficient. The partial pressure driving force is the difference of the vapor pressure of water at the surface temperature $p_{w,s}^*$ and the partial pressure of water in the cloud, $p_{w,c}$. (The water partial pressure in the cloud is the minimum of: (a) the water mole fraction times the ambient pressure; or (b) the water vapor pressure at the cloud temperature ($p_{w,c}^*$).) The natural convection coefficient is based on the heat transfer coefficient and the analogy between the Sherwood number (Sh) and the Nusselt number (Nu) suggested by Bird et al. (1960)

$$Sh = Nu = 0.14 \, (Gr \, Sc)^{1/3} = \frac{F_n L}{D'} \left[\frac{MW}{\rho} \right] \tag{V.47}$$

If the Schmidt number is taken as 0.6, and $\left[\frac{\mu}{T \, MW} \right]$ is estimated to be 2.2×10^{-9} in mks units,

$$F_n = 9.9 \times 10^{-3} \left[\left[\frac{\rho}{MW} \right]^2 \Delta T \right]^{1/3} \tag{V.48}$$

For forced convection, Treybal (1980) suggests that the Stanton number for mass transfer St_M and the Stanton number for heat transfer St_H are related by

$$St_M = St_H \left[\frac{Pr}{Sc}\right]^{2/3} = 1.15\ St_H \tag{V.49}$$

or,

$$F_f = \frac{20.7\ h_0}{MW\ C_p} \tag{V.50}$$

The overall mass transfer coefficient F_0 is the larger of the natural and forced convection coefficients.

For the case when the primary (liquid) source emission rate E(t) is larger than the atmospheric takeup rate $Q_{*max}\pi R_p^2$, Equations (V.32), (V.34), (V.35), and (V.36) are integrated for the mass, concentration, and enthalpy of the gas blanket along with an appropriate equation of state (i.e. relationship between enthalpy and temperature and between temperature and density).

When the emission rate is not sufficient to form a gas blanket, the flux of contaminant is not determined by the maximum atmospheric takeup rate. Consider the boundary layer formed by the emission of gas into the atmosphere above the primary source. If the source is modeled to have a uniform width 2b and entrain no air along the sides of the layer, the balance on the total material ($\rho_L u_L H_L$) in a differential slice of the layer is

$$\frac{d}{dx}\left[\rho_L u_L H_L\right] = \rho_a w_e + \left(\frac{Q_*}{w_c}\right)_p \tag{V.51}$$

where w_e is the vertical rate of of air entrainment into the layer given by Equation (V.83), ρ_L is the average density of the slice, and $(Q_*/w_c)_p$ is the total flux of gas from the primary source. The balance on the mass flow rate of contaminant ($w_c \rho_L u_L H_L$) at any ($x - x_{up}$) is

$$c_{c,L} u_L H_L = Q_*(x - x_{up}) \tag{V.52}$$

With an equation of state to relate $c_{c,L}$ and ρ_L, Equation (V.51) is integrated from the upwind edge of the source ($x = x_{up}$) to the downwind edge ($x = L + x_{up}$).

In order to generate the initial conditions for the downwind dispersion calculations, the maximum concentration c_c and the vertical dispersion parameter S_z are needed. Since Equations (V.51) and (V.52) are written for a vertically averaged layer, consider the vertical average of the power law distribution. The height of the layer H_L is the height to some concentration level, say 10% of the maximum. Although strictly a function of α, this value is modeled as

$$H_L = \delta_L \, H_{EFF} \tag{V.53}$$

where H_{EFF} is the effective height defined by Equation (V.79) and $\delta_L = 2.15$. The vertically averaged concentration $c_{c,L}$ is defined by

$$c_{c,L} H_L = \int_0^\infty c \, dz \tag{V.54}$$

Similarly, the effective transport velocity u_L is defined by

$$c_{c,L} u_L H_L = \int_0^\infty c u_x \, dz \tag{V.55}$$

With Equation (V.53) and defining relations for H_{EFF} and u_{EFF} (Equations (V.79) and (V.93), respectively), it follows that

$$c_c = \delta_L c_{c,L} \tag{V.56}$$

$$u_L H_L = \delta_L \left(\frac{u_0 z_0}{1 + \alpha} \right) \left(\frac{S_z}{z_0} \right)^{1+\alpha} \tag{V.57}$$

and

$$\delta_L w_e' = w_e \tag{V.58}$$

where w_e' is given by Equation (V.83).

Maximum Atmospheric Takeup Rate

The maximum atmospheric takeup rate is determined as the largest takeup rate which satisfies Equations (V.51) and (V.52), and the maximum concentration of contaminant at the downwind edge of the source is the source contaminant concentration $(c_c)_s$. Combining Equations (V.51) and (V.52) and assuming adiabatic mixing of ideal gases with equal molal heat capacities (i.e. $(\rho - \rho_a)/c_c = \gamma =$ constant), the maximum takeup flux is modeled as

$$Q_{*max} = (c_c)_s \frac{k u_*(1 + \alpha)}{\hat{\phi}} \left[\frac{\delta_L}{\delta_L - 1} \right] \tag{V.59}$$

where

$$\frac{1}{\hat{\phi}} = \frac{1}{L} \int_0^L \frac{dx}{\phi(Ri_*)} \tag{V.60}$$

and $\phi(Ri_*)$ is defined (for $\rho \ge \rho_a$) by Equation (V.76).

An upper bound for the atmospheric takeup flux can be associated with the condition where the source begins to spread as a gravity intrusion against the approach flow. In water flume experiments, Britter (1980) measured the upstream and lateral extent of a steady-state plume from a circular source as a function of Ri_*. A significant upstream spread was obtained for $Ri_* > 32$, and lateral spreading at the center of the source was insignificant for $Ri_* < 8$. The presence of any significant lateral spreading represents a lower bound on the conditions of the maximum takeup flux.

The integral of Equation (V.60) is evaluated using a local Richardson number

$$Ri_*(x) = \varsigma(x - x_{up})^{\frac{1}{1+\alpha}} \tag{V.61}$$

where

$$\varsigma = g\left(\frac{\rho - \rho_a}{\rho_a}\right)\frac{z_0}{u_*^2}\,\frac{\Gamma\left(\frac{1}{1+a}\right)}{1+\alpha}\left[\frac{ku_*(1 + \alpha)}{\phi_c}\left(\frac{1 + \alpha}{u_0 z_0}\right)\left(\frac{\delta_L}{\delta_L - 1}\right)\right]^{\frac{1}{1+\alpha}} \tag{V.62}$$

and ϕ_c is 3.1 (corresponding to $Ri_* = 20(8 < Ri_* < 32)$). Equation (V.60) then becomes

$$\frac{1}{\hat{\phi}} = \frac{1}{L}\int_0^L \frac{dx}{0.88 + 0.099\,\xi^{1.04} x^{\frac{1.04}{1+\alpha}}} \tag{V.63}$$

Equation (V.63) can be solved analytically (Gradshteyn and Ryzhik, 1980). With $\hat{\phi}$ determined, Q_{*max} can be determined with Equation (V.59).

Simulation of Transient Gas Releases

For simulating steady-state releases, the transient source calculation is carried out until the source characteristics no longer change (significantly) with time. The maximum centerline concentration c_c, the horizontal and vertical dispersion parameters S_y and S_z, the half width b, and if necessary, the enthalpy h are used as initial conditions for the downwind calculation specified in a transient spill.

Transient releases are modeled as a series of pseudo-steady-state releases. Consider a series of observers traveling with the wind over the transient gas source described above; each observer originates from the point which corresponds with the maximum upwind extent of the gas blanket ($x = -R_{max}$). The observer velocity is u_{EFF} (Equation (V.93)), the average transport velocity of the gas. Since, in general, the value of u_{EFF} may differ from observer to observer, observer separation is not guaranteed. For a neutrally buoyant cloud, u_{EFF} is a function of downwind distance alone, and observer separation is guaranteed. Following Colenbrander (1980), the observer velocity is modeled as:

$$u_i(x) = \frac{u_0}{\Gamma\left(\frac{1}{1+\alpha}\right)} \left(\frac{S_{z_{0m}}}{z_0}\right)^{\alpha} \left[\frac{x + R_{max}}{\frac{\sqrt{\pi}}{2} R_m + R_{max}}\right]^{\alpha/(1+\alpha)} \tag{V.64}$$

where $S_{z_{0m}}$ is the value of S_{z_0} when the averaged source rate ($\pi R^2 Q_*$) is a maximum, and the subscript i denotes observer i. Noting that $u_i(x) = dx_i/dt$, observer position and velocity are determined.

A pseudo-steady-state approximation of the transient source is determined as each observer passes over the source. If t_{up_i} and t_{dn_i} denote the times when observer i encounters the upwind and downwind edges of the source respectively, then the source fetch seen by observer i is:

$$L_i = x_{up_i} - x_{dn_i} \tag{V.65}$$

The width of the source $2B_i'(t)$ is defined by

$$B_i'^2(t) = R^2(t) - x_i^2(t) \tag{V.66}$$

Then the gas source area seen by observer i is

$$2L_i b_i = 2\int_{t_{up_i}}^{t_{dn_i}} B_i' u_i \, dt \tag{V.67}$$

where $2b_i$ is the average width.

The takeup rate of contaminant $2(Q_* Lb)_i$ is calculated as

$$2(Q_* Lb)_i = 2\int_{t_{up_i}}^{t_{dn_i}} Q_* B_i' u_i \, dt \tag{V.68}$$

The total mass flux rate from the source is

$$2(\rho_L u_L H_L b)_i = 2\int_{t_{up_i}}^{t_{dn_i}} \left[\rho_a w'_e + \left[\frac{Q_*}{w_c}\right]\right] B'_i u_i\,dt \qquad \text{(V.69)}$$

The average composition of the layer can now be determined at each $x = x_{up}$ over the source. The enthalpy of the layer is given by

$$2(h_L \rho_L u_L H_L b)_i = 2\int_{t_{up_i}}^{t_{dn_i}} h \left[\frac{Q_*}{w_c}\right] B'_i u_i\,dt \qquad \text{(V.70)}$$

(due to the choice of the reference temperature as the ambient temperature). With a suitable equation of state relating enthalpy, temperature, and density, the source can be averaged for each observer. After the average composition of the layer is determined at the downwind edge, an adiabatic mixing calculation is performed for gas of the layer average concentration and the ambient air. This calculation determines the function between density and concentration relationship for the remainder of the simulation if the calculation is adiabatic; it represents the adiabatic mixing condition if heat transfer is included in the downwind simulation.

For each of several observers released successively from $x = -R_{max}$, the observed dimensions L and b, the downwind edge of the source x_{dn}, the average vertical dispersion coefficient S_z, the average takeup flux Q_*, the centerline concentration c_c, and if applicable, the average enthalpy h_L is determined. A steady-state calculation is then made for each observer. The distribution parameters for any specified time t_s are determined by locating the position of the series of observers at time t_s, i.e. $x_i(t_s)$. The corresponding concentration distribution is then computed from the assumed profiles.

Steady-State Downwind Dispersion

The model treats dispersion of gas entrained into the wind field from an idealized, rectangularly shaped source of width 2b and length L. The circular source cloud is represented as an equivalent area rectangle ($\pi R^2 = 2bL$) with equivalent fetch ($L = 2R$). Similarity forms for the concentration profiles represent the plume as a horizontally homogeneous center section with Gaussian concentration profile edges:

$$c(x,y,z) = c_c(x) \exp\left[-\left(\frac{|y| - b(x)}{S_y(x)}\right)^2 - \left(\frac{z}{S_z(x)}\right)^{1+\alpha}\right]$$

for $|y| > b$

$$= c_c(x) \exp\left[- \left(\frac{z}{S_z(x)} \right)^{1+\alpha} \right] \qquad \text{for } |y| \leq b \tag{V.71}$$

A power law wind velocity profile is assumed

$$u_x = u_0 \left(\frac{z}{z_0} \right)^{\alpha} \tag{V.72}$$

where the value of α is determined by a weighted least-squares fit of the logarithmic profile

$$u_x = \frac{u_*}{k} \left[\ln \left(\frac{z + z_R}{z_R} \right) - \Psi \left(\frac{z}{\lambda} \right) \right] \tag{V.73}$$

Functional forms for Ψ and typical values of α are given in Table V.1 for different Pasquill stability categories. With these profiles, the parameters of Equation (V.71) are constrained by ordinary differential equations.

TABLE V.1
TYPICAL ATMOSPHERIC BOUNDARY LAYER STABILITY AND WIND PROFILE CORRELATIONS

Pasquill Stability Category	Monin-Obukhov Length (λ) as a Function of Surface Roughness z_R(m)[1]	Typical Power Law Exponents α in Eqn. (V.72)	Corrections to Logarithmic Profiles as Given by Businger et al. (1971) Ψ in Eqn. (V.73)
A	$-11.4\ z_R^{0.10}$	0.108	$\Psi = 2 \ln \left[\frac{1+a}{2}\right] + \ln \left[\frac{1+a^2}{2}\right] - 2 \tan^{-1}(a) + \pi/2$ with $a = (1 - 15(z/\lambda))^{1/4}$
B	$-26.0\ z_R^{0.17}$	0.112	
C	$-123\ z_R^{0.30}$	0.120	
D	∞	0.142	$\Psi = 0$
E	$123\ z_R^{0.30}$	0.203	$\Psi = -4.7\ (z/\lambda)$
F	$26.0\ z_R^{0.17}$	0.253	

[1] Curve fit of data from Pasquill (1974).

Vertical Dispersion

The vertical dispersion parameter S_z is determined by requiring that it satisfy the diffusion equation

$$u_x \frac{\partial c}{\partial x} = \frac{\partial}{\partial z} K_z \frac{\partial c}{\partial z} \qquad (V.74)$$

with the vertical turbulent diffusivity given by

$$K_z = \frac{k u_* z}{\phi(Ri_*)} \qquad (V.75)$$

The function $\phi(Ri_*)$ is a curve fit of laboratory data for vertical mixing in stably density-stratified fluid flows ($Ri_* > 0$) reported by Kantha et al. (1977), Lofquist (1960), and McQuaid (1976). For $Ri_* < 0$, the function $\phi(Ri_*)$ was taken from Colenbrander and Puttock (1983) and modified so the passive limit of the two functions agree as follows:

$$\begin{aligned} \phi(Ri_*) &= 0.88 + 0.099\,Ri_*^{1.04} && Ri_* \ge 0 \\ &= 0.88/(1 + 0.65\,|Ri_*|^{0.6}) && Ri_* < 0 \end{aligned} \tag{V.76}$$

The friction velocity is calculated using Equation (V.73) from a known velocity u_0 at a specific height z_0. Combining the assumed similarity forms for concentration and velocity, Equations (V.71), (V.72), (V.74), and (V.75) give

$$\frac{d}{dx}\left[\left(\frac{u_0 z_0}{1 + \alpha}\right)\left(\frac{S_z}{z_0}\right)^{1+\alpha}\right] = \frac{k u_*(1 + \alpha)}{\phi(Ri_*)} \tag{V.77}$$

where the Richardson number Ri_* is computed as

$$Ri_* = g\left(\frac{\rho - \rho_a}{\rho_a}\right)\frac{H_{EFF}}{u_*^2} \tag{V.78}$$

and the effective cloud depth is defined as

$$H_{EFF} = \frac{1}{c_c}\int_0^\infty c\,dz = \Gamma\left(\frac{1}{1 + \alpha}\right)\frac{S_z}{1 + \alpha} \tag{V.79}$$

Equation (V.77) can be viewed as a volumetric balance on a differential slice downwind of the source. A mass balance over the same slice gives,

$$\frac{d}{dx}\left[\rho_L u_L H_L\right] = \rho_a w_e \tag{V.80}$$

which is Equation (V.51) without the source term. With Equations (V.57) and (V.58), this becomes

$$\frac{d}{dx}\left[\rho_L u_{EFF} H_{EFF}\right] = \rho_a w'_e \tag{V.81}$$

Assuming adiabatic mixing of ideal gases with equal constant molal heat capacities (i.e. $(\rho - \rho_a)/c_c = \text{constant}$) and using the contaminant mass balance, the mass balance becomes

$$\frac{d}{dx}\left[u_{EFF} H_{EFF} \right] = w'_e \tag{V.82}$$

which leads to

$$w'_e = \frac{w_e}{\delta_L} = \frac{k u_*(1 + \alpha)}{\phi(Ri_*)} \tag{V.83}$$

Equations (V.81) and (V.83) are combined to give

$$\frac{d}{dx}\left[\rho_L u_{EFF} H_{EFF} \right] = \frac{\rho_a k u_*(1 + \alpha)}{\phi(Ri_*)} \tag{V.84}$$

Generalizing, Equation (V.84) is assumed to apply when $(\rho_a - \rho_c)/c$ is not constant.

When heat transfer from the surface is present, vertical mixing will be enhanced by the convection turbulence due to heat transfer. Zeman and Tennekes (1977) model the resulting vertical turbulent velocity as

$$\frac{w}{u_*} = \left[1 + \frac{1}{4}\left(\frac{w_*}{u_*} \right)^2 \right]^{1/2} \tag{V.85}$$

where w_* is the convective scale velocity described as

$$\left(\frac{w_*}{u_*} \right)^2 = \left[\frac{gh}{u_* \bar{u}} \frac{\left(T_s - T_{c,L} \right)}{T_{c,L}} \right]^{2/3} \tag{V.86}$$

If $\bar{u}$ is is evaluated at H_{EFF},

$$\frac{w}{u_*} = \left(1 + \frac{1}{4} Ri_T^{2/3} \right)^{1/2} \tag{V.87}$$

where

$$Ri_T = g \left(\frac{T_s - T_{c,L}}{T_{c,L}} \right) \frac{H_{EFF}}{u_* u_0} \left(\frac{z_0}{H_{EFF}} \right)^{\alpha} \tag{V.88}$$

and $T_{c,L}$ is the temperature obtained from the energy balance of Equations (V.103) and (V.104). Equation (V.84) is modified to account for this enhanced mixing by

$$\frac{d}{dx}\left[\rho_L u_{EFF} H_{EFF}\right] = \frac{\rho_a kw(1+\alpha)}{\phi(Ri_*')} \tag{V.89}$$

where $Ri_*' = Ri_* \left[\frac{u_*}{w}\right]^2$.

Although derived for two-dimensional dispersion, Equation (V.89) is extended for application to a denser-than-air gas plume which spreads laterally as a density intrusion:

$$\frac{d}{dx}\left[\rho_L u_{EFF} H_{EFF} B_{EFF}\right] = \frac{\rho_a kw(1+\alpha)}{\phi(Ri_*')} B_{EFF} \tag{V.90}$$

where the plume effective half width is defined by

$$B_{EFF} = b + \frac{\sqrt{\pi}}{2} S_y \tag{V.91}$$

and determined using the gravity intrusion relation

$$\frac{dB_{EFF}}{dt} = C_E \left[g\left(\frac{\rho - \rho_a}{\rho_a}\right) H_{EFF}\right]^{1/2} \tag{V.92}$$

The average transport velocity in the plume is defined by

$$u_{EFF} = \frac{\int_0^\infty cu_x dz}{\int_0^\infty cdz} = u_0 \left(\frac{S_z}{z_0}\right)^\alpha \Big/ \Gamma\left(\frac{1}{1+\alpha}\right) \tag{V.93}$$

and the lateral spread of the cloud is modeled as

$$\frac{dB_{EFF}}{dx} = \frac{1}{u_{EFF}} \frac{dB_{EFF}}{dt}$$

$$= C_E \left[\frac{g z_0 \Gamma^3 \left(\frac{1}{1+\alpha}\right)}{u_0^2 (1 + \alpha)} \right]^{1/2} \left(\frac{\rho - \rho_a}{\rho_a} \right)^{1/2} \left(\frac{S_z}{z_0} \right)^{\left(\frac{1}{2} - \alpha\right)} \qquad \text{(V.94)}$$

Horizontal Dispersion

The crosswind similarity parameter $S_y(x)$ is also determined by requiring that it satisfy the diffusion equation

$$u_x \frac{\partial c}{\partial x} = \frac{\partial}{\partial y} \left[K_y \frac{\partial c}{\partial y} \right] \qquad \text{(V.95)}$$

with the horizontal turbulent diffusivity given by

$$K_y = K_0 u_x B_{EFF}^{\gamma_1} \qquad \text{(V.96)}$$

For a passive plume ($b = 0$), $S_y = \sqrt{2}\sigma_{y,a}$ where $\sigma_{y,a}$ is the similarity parameter correlated by Pasquill (1974) in the form $\sigma_{y,a} = \delta_y x^{\beta_y}$ where δ_y and β_y are functions of the Pasquill stability category and the averaging time. Furthermore, Equations (V.95) and (V.96) require that

$$\sigma_{y,a} \frac{d\sigma_{y,a}}{dx} = K_0 \, B_{EFF}^{\gamma_1} \qquad \text{(V.97)}$$

where $\gamma_1 = 2 - 1/\beta_y$ and $K_0 = \frac{2\beta_y}{\pi} (\delta_y \sqrt{\pi/2})^{1/\beta_y}$. Then,

$$S_y \frac{dS_y}{dx} = \frac{4\beta_y}{\pi} B_{EFF}^2 \left[\frac{\delta_y \sqrt{\pi/2}}{B_{EFF}} \right]^{1/\beta_y} \qquad \text{(V.98)}$$

where Equation (V.98) is also assumed applicable for determining S_y when b is not zero.

At the downwind distance x_t where $b = 0$, the crosswind concentration profile is assumed Gaussian with S_y given by

$$S_y = \sqrt{2}\, \delta \, (\underset{y}{x} + x_v)^{\beta} \qquad \text{(V.99)}$$

where x_v is a virtual source distance determined as

$$S_y(x_t) = \sqrt{2}\, \delta_y (x_t + x_v)^{\beta} \qquad \text{(V.100)}$$

The gravity spreading calculation is terminated for $x > x_t$.

Averaging Time

The observed dispersion of a gas plume is a function of the observation (averaging) time. Herein we have assumed:

- The most important influence of averaging time on observed plume properties is associated with plume meander.
- For ground-level releases, vertical plume meander is much smaller than horizontal plume meander.

Gifford (1960) showed that the ratio of maximum measured concentrations based on different averaging times (t_a and t_p) was given by

$$\frac{c_c(t_p)}{c_c(t_a)} = \left[\frac{t_a}{t_p} \right]^{0.2}$$

for t_a/t_p up to about 200, when source and receptor were at the same level. Spicer and Havens (1988) proposed that

$$\frac{\sigma_{y,a}(t_a)}{\sigma_{y,a}(t_{a,i})} \simeq \left[\frac{t_a}{t_{a,i}} \right]^{0.2}$$

where $t_{a,i}$ was suggested to be an "instantaneous" averaging time representing the smallest time scale associated with plume meander. (Practically, $t_{a,i}$ represents a nonzero averaging time associated with the puff coefficient $\sigma_{y,a}(t_{a,i})$.) Based on correlations from Gifford (1976) and Slade (1968), Spicer and Havens (1988) presented the following dispersion coefficient correlations for stabilities A through F:

$$\sigma_{y,a}(x;t_a) = 0.423 \left[\frac{t_a^*}{600\ \mathrm{s}} \right]^{0.2} x^{0.9} \qquad \mathrm{A} \tag{V.101a}$$

$$\sigma_{y,a}(x;t_a) = 0.313 \left[\frac{t_a^*}{600\ \mathrm{s}} \right]^{0.2} x^{0.9} \qquad \mathrm{B} \tag{V.101b}$$

$$\sigma_{y,a}(x;t_a) = 0.210 \left[\frac{t_a^*}{600\ \mathrm{s}} \right]^{0.2} x^{0.9} \qquad \mathrm{C} \tag{V.101c}$$

$$\sigma_{y,a}(x;t_a) = 0.136 \left[\frac{t_a^*}{600\ \mathrm{s}} \right]^{0.2} x^{0.9} \qquad \mathrm{D} \tag{V.101d}$$

$$\sigma_{y,a}(x;t_a) = 0.102 \left[\frac{t_a^*}{600\ \mathrm{s}} \right]^{0.2} x^{0.9} \qquad \mathrm{E} \tag{V.101e}$$

$$\sigma_{y,a}(x;t_a) = 0.0674 \left[\frac{t_a^*}{600\ s} \right]^{0.2} x^{0.9} \qquad F \qquad (V.101f)$$

where $t_a^* = \max(t_a,\ t_{a,i})$ and t_a is the desired averaging time. Values of $t_{a,i}$ are shown in Table V.2.

Table V.2. Values of $t_{a,i}$ for Use in Eq. (V.101)

Pasquill Stability	$t_{a,i}(s)$
A	18.4
B	18.4
C	18.4
D	18.3
E	11.4
F	4.6

Values for δ_y and β_y are then determined by Equation (V.101). (For example, for an averaging time of 60 s for F stability, $t_a^* = 60$ s, $\delta_y = 0.0425$, and $\beta_y = 0.9$.)

(Thermal) Energy Balance

For some simulations of cryogenic gas releases, heat transfer to the plume in the downwind dispersion calculation may be important, particularly in low wind conditions. The source calculation determines a gas/air mixture initial condition for the downwind dispersion problem. Air entrained into the plume is assumed to mix adiabatically. Heat transfer to the plume downwind of the source adds additional heat. This added heat per unit mass D_h is determined by an energy balance on a uniform cross-section as

$$\frac{d}{dx} \left[D_h \rho_L u_{EFF} H_{EFF} \right] = q_s / \delta_L \qquad (V.102)$$

where q_s is determined by Equation (V.37) along with the desired method of calculating h_0. Equation (V.102) is applied when $b = 0$ and is generalized to

$$\frac{d}{dx} \left[D_h \rho_L u_{EFF} H_{EFF} B_{EFF} \right] = q_s B_{EFF} / \delta_L \qquad (V.103)$$

when $b > 0$. Since the average density of the layer ρ_L cannot be determined until the temperature (i.e. D_h) is known, a trial and error procedure is required.

Closure

For a steady plume, the centerline concentration c_c is determined from the material balance

$$E = \int_0^\infty \int_{-0}^\infty c u_x \, dy dz = 2 c_c \left(\frac{u_0 z_0}{1 + \alpha} \right) \left(\frac{S_z}{z_0} \right)^{1+\alpha} B_{EFF} \qquad (V.104)$$

where E is the plume source strength.

Equations (V.76), (V.78), (V.79), (V.85)-(V.91), (V.93), (V.94), (V.98)-(V.100), and (V.102)-(V.104) are combined with an equation of state relating cloud density to gas concentration and temperature and are solved simultaneously to predict S_z, S_y, c_c, and b as functions of downwind distance beginning at the downwind edge of the gas source.

Correction for Along-Wind Dispersion

Following Colenbrander (1980), an adjustment to c_c is applied to account for dispersion parallel to the wind direction. The calculated centerline concentration $c_c(x)$ is considered to have resulted from the release of successive planar puffs of gas ($c_c(x)\Delta x$) without any dispersion in the x-direction. If it is assumed that each puff diffuses in the x-direction as the puff moves downwind independently of any other puff and that the dispersion is one-dimensional and Gaussian, the x-direction concentration dependence is given by

$$c_c' (x; x_{p_i}) = \frac{c_c(x_{p_i}) \Delta x_i}{\sqrt{2\pi}\, \sigma_{x,a}} \exp \left[-\frac{1}{2} \left[\frac{x - x_{p_i}}{\sigma_x} \right]^2 \right] \qquad (V.105)$$

where x_{p_i} denotes the position of the puff center due to observer i.

After Beals (1971), the x-direction dispersion coefficient $\sigma_{x,a}$ is assumed to be a function of distance from the downwind edge of the gas source ($X = x - x_0$) and atmospheric stability given by

$$\begin{aligned} \sigma_{x,a}(X) &= 0.02\, X^{1.22} && \text{unstable, } x \geq 130 \text{ m} \\ &= 0.04\, X^{1.14} && \text{neutral, } x \geq 100 \text{ m} \\ &= 0.17\, X^{0.97} && \text{stable, } x \geq 50 \text{ m} \end{aligned} \qquad (V.106)$$

where ($X = x - x_0$) and $\sigma_{x,a}$ are in meters. The concentration at x is then determined by superposition, i.e., the contribution to c_c at a given x from neighboring puffs is added to give an x-direction corrected value of c_c'. For N observers,

$$c_c'(x) = \sum_{i=1}^{N} \frac{c_c(x_{p_i})}{\sqrt{2\pi}\, \sigma_{x,a}} \exp \left[-\frac{1}{2} \left[\frac{x - x_{p_i}}{\sigma_{x,a}} \right]^2 \right] \Delta x_i \qquad (V.107)$$

and for large N,

$$c_c'(x) = \frac{1}{\sqrt{2\pi}} \int_0^\infty \frac{c_c(\xi)}{\sigma_{x,a}(\xi - \xi_0)} \exp\left[-\frac{1}{2} \left[\frac{x - \xi}{\sigma_{x,a}(\xi - \xi_0)} \right]^2 \right] d\xi \qquad \text{(V.108)}$$

The corrected centerline concentration c_c' is used in the assumed profiles in place of c_c, along with the distribution parameters S_y, S_z, and b.

VI. CONCLUSIONS AND RECOMMENDATIONS

DEGADIS was designed to model the atmospheric dispersion of ground-level, area-source dense gas (or aerosol) clouds released with zero (initial) momentum into an atmospheric boundary layer over flat, level terrain. DEGADIS describes the dispersion processes which accompany the ensuing gravity-driven flow and entrainment of the gas into the atmospheric boundary layer.

A jet-plume model has been interfaced with DEGADIS to provide for prediction of the trajectory and dilution of vertically oriented gas or aerosol jets. When the jet plume returns to ground level, DEGADIS predicts the ensuing ground-level plume dispersion. The jet-plume model provides for

- automatic adjustment of integration step-size (using the Runge-Kutta-Gill method as in DEGADIS),
- elliptical plume shape (cross-section), with air entrainment specified consistent with the Pasquill-Gifford plume dispersion coefficient representation of atmospheric turbulent entrainment,
- application to scenarios where the plume returns to ground level or remains aloft, and
- ground reflection when the plume (lower) boundary reaches ground level.

It is recommended that the following limitations and cautions be observed when using the jet-plume and DEGADIS models.

- The jet-plume model presently provides only for vertically oriented releases.
- Both models assume an unobstructed atmospheric flow field. The jet-plume model assumes a logarithmic wind profile, and the DEGADIS model assumes a power law wind profile which is consistent with the logarithmic wind profile. Application of the models should be limited to releases where the depth of the dispersing layer is much greater than the height of the surface roughness elements.
- Averaging time can be user-specified. There are three important time scales for the typical dispersion application: the averaging time input to the model (t_{av}), the time scale assumed for specification of the concentration/hazard relation (t_{haz}), and the time required for the contaminant material to be taken up by

the atmosphere (t_{rel}). The averaging time t_{av} influences the rate of lateral spread (dispersion) due to atmospheric turbulence (δ_y). For an elevated release which does not return to the ground, t_{rel} is the duration of the release. For a transient ground-level release or for an elevated release which returns to the ground, t_{rel} is the duration of the secondary source blanket in DEGADIS. For a steady-state release (large t_{rel}), t_{av} should be set to t_{haz}. (For example, if t_{haz} is based on toxicological data for one hour exposure, then $t_{haz} = t_{av} = 3600$ s.) For a transient release, t_{av} should be set to the minimum of t_{haz} or t_{rel}. If $t_{haz} \leq t_{rel}$, $t_{av} = t_{haz}$. If $t_{rel} < t_{haz}$, $t_{av} = t_{rel}$, and the concentration time history at a given position should be averaged over t_{haz} to determine the hazard.

REFERENCES

Albertson M. L., Y. B. Dai, R. A. Jenson, and H. Rouse, "Diffusion of Submerged Jets," Transactions of American Society of Civil Engineers, 115, (1950).

Astleford, W. J., T. B. Morrow, and J. C. Buckingham, "Hazardous Chemical Vapor Handbook for Marine Tank Vessels," Final Report to U.S. Coast Guard, CG-D-12-83, USCG HQ, Washington, DC, April, 1983.

Atwood, J. D., "Ammonia Pipeline Failure Near Enid, Oklahoma," Paper No. 45f, 82nd American Institute of Chemical Engineers, Atlantic City, New Jersey, August 30-September 1, 1976.

Batchelor, G. K., An Introduction to Fluid Dynamics, Cambridge University Press, Cambridge, UK, 1967.

Beals, G. A., "A Guide to Local Dispersion of Air Pollutants," Air Weather Service Technical Report 214, April, 1971.

Bird, R. B., W. E. Stewart, and E. N. Lightfoot, Transport Phenomena, John Wiley and Sons, New York, 1960.

Briggs, G. A., "Plume Rise," AEC Critical Review Series, USAEC Division of Technical Information Extension, Oak Ridge, Tennessee, 1969.

Britter, R. E., "The Ground Level Extent of a Negatively Buoyant Plume in a Turbulent Boundary Layer," Atmospheric Environment, 14, 1980.

Britter, R. E., unpublished monograph, 1980.

Businger, J. A., J. C. Wyngaard, Y. Izumi, and E. F. Bradley, "Flux-Profile Relationships in the Atmospheric Surface Layer," Journal of the Atmospheric Sciences, 28, March, 1971.

Carnahan, B., H. A. Luther, and J. O. Wilkes, Applied Numerical Methods, John Wiley and Sons, 1969.

Chassaing, P., J. George, A. Claria, and F. Sananes, "Physical Characteristics of Subsonic Jets in a Cross-Stream," Journal of Fluid Mechanics, 62, 41, 1974.

Chiang, Hsu-Cherny and B. L. Sill, "Entrainment Models and their Application to Jets in a Turbulent Cross Flow," Atmospheric Environment, 19, 9, 1425-1438, 1985.

Colenbrander, G. W., "A Mathematical Model for the Transient Behavior of Dense Vapor Clouds," 3rd International Symposium on Loss Prevention and Safety Promotion in the Process Industries, Basel, Switzerland, 1980.

Colenbrander, G. W. and J. S. Puttock, "Dense Gas Dispersion Behavior: Experimental Observations and Model Developments," International Symposium on Loss Prevention and Safety Promotion in the Process Industries, Harrogate, England, September, 1983.

Cude, I. L., "Dispersion of Gases Vented to Atmosphere from Relief Valves," The Chemical Engineer, October, 1974.

Davidson, G. A., "A Discussion of Schatzmann's Integral Plume Model from a Control Volume Viewpoint," Journal of Climate and Applied Meteorology, 25, 858-867, 1986.

Emerson, M. C., "A New 'Unbounded' Jet Dispersion Model," Fifth International Symposium on Loss Prevention and Safety Promotion in the Process Industries, Cannes, France, September, 1987.

Gifford, F., "Peak to Average Concentration Ratios According to a Fluctuating Plume Dispersion Model," International Journal of Air Pollution, 3(4), 253-260, 1960.

Gifford, F., "Turbulent Diffusion-Typing Schemes: A Review,"Nuclear Safety, 17(1), 68-86, 1976.

Gradshteyn, I. S. and I. M. Ryzhik, Table of Integrals, Series, and Products, corrected and enlarged edition, Academic Press, New York, 1980.

Guinnup, D. W., personal communication with reference to EPA Relief Valve Discharge (RVD) model, January, 1989.

Havens, Jerry, "A Dispersion Model for Elevated Dense Gas Jet Chemical Releases," Volumes I and II, EPA-450/4-88-006, U.S. Environmental Protection Agency, April, 1988.

Havens, J. A. and T. O. Spicer, "Development of an Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures," Final Report to U.S. Coast Guard, CG-D-23-85, USCG HQ, Washington, DC, May, 1985.

Hoot, T. G., R. N. Meroney, and J. A. Peterka, "Wind Tunnel Tests of Negatively Buoyant Plumes," Report CER73-74TGH-RNM-JAP-13, Fluid Dynamics and Diffusion Laboratory, Colorado State University, October, 1973.

Kaimal, J. C., J. C. Wyngaard, D. A. Haugen, O. R. Cote, and Y. Izumi, "Turbulence Structure in the Convective Boundary Layer," J. Atmos. Sci., 33, 1976.

Kamotani, Yasuhiro and Isaac Greber, "Experiments on a Turbulent Jet in a Cross Flow," AIAA Journal, Vol. 10, No. 11, November, 1972.

Kantha, H. L., O. M. Phillips, and R. S. Azad, "On Turbulent Entrainment at a Stable Density Interface," Journal of Fluid Mechanics, 79, 1977, pp. 753-768.

Keffer, J. F. and W. D. Baines, "The Round Turbulent Jet in a Cross-Wind," Journal of Fluid Mechanics, 15, 4, 481-496, 1963.

Koopman, R. P. et al., "Description and Analysis of Burro Series 40-m^3 LNG Spill Experiments," Lawrence Livermore National Laboratories Report UCRL-53186, August 14, 1981.

List, E. J., "Mechanics of Turbulent Buoyant Jets and Plumes," in W. Rodi, Turbulent Buoyant Jets and Plumes, Pergamon Press, Oxford, 1982.

Lofquist, Karl, "Flow and Stress Near an Interface Between Stratified Liquids," Physics of Fluids, 3, No. 2, March-April, 1960.

McAdams, W. H., Heat Transmission, McGraw-Hill, New York, 1954.

McQuaid, James, "Some Experiments on the Structure of Stably Stratified Shear Flows," Technical Paper P21, Safety in Mines Research Establishment, Sheffield, UK, 1976.

Moller, J. S., H. Schroeder, and I. S. Hansen, "An Integral Model of the Buoyant Surface Jet and Plume in a Cross Current and Wind," presented at the Third International Symposium on Stratified Flows, California Institute of Technology, Pasadena, California, February, 1987.

Morrow, T. B., "Analytical and Experimental Study to Improve Computer Models for Mixing and Dilution of Soluble Hazardous Chemicals," Final Report, Contract DOT-CG-920622-A with Southwest Research Institute, U.S. Coast Guard, August, 1982.

Morrow, T. B., "Investigation of the Hazards Posed by Chemical Vapors Released in Marine Operations--Phase II," Southwest Research Institute Project Report 05-5686, July, 1985.

Ooms, G., "A New Method for the Calculation of the Plume Path of Gases Emitted by a Stack," Atmospheric Environment, 6, 899-909, 1972.

Ooms, G., A. P. Mahieu, and F. Zelis, "The Plume Path of Vent Gases Heavier than Air," First International Symposium on Loss Prevention and Safety Promotion in the Process Industries, (C. H. Buschman, Editor), Elsevier Press, 1974.

Pasquill, F., Atmospheric Diffusion, 2nd edition, Halstead Press, New York, 1974.

Pratte, B. D. and W. D. Baines, "Profiles of the Round Turbulent Jet in a Cross Flow," ASCE--Journal of the Hydraulics Division, 92, HY6, 5556, 1967.

Richards, J. M., "Experiments on the Motion of an Isolated Cylindrical Thermal Through Unstratified Surroundings," International Journal of Air and Water Pollution, 7, 1963.

Schatzmann, M., "The Integral Equations for Round Buoyant Jets in Stratified Flows," ZAMP, 29, 608-630, 1978.

Schatzmann, M. and A. P. Policastro, "An Advanced Integral Model for Cooling Tower Plume Dispersion," Atmospheric Environment, 18, 4, 663-674, 1984.

Seinfeld, J. H., Atmospheric Chemistry and Physics of Air Pollution, John Wiley and Sons, New York, 1986.

Simpson, J. E. and R. E. Britter, "The Dynamics of the Head of a Gravity Current Advancing over a Horizontal Surface," Journal of Fluid Mechanics, 94, Part 3, 1979.

Slade, D. H., "Meteorology and Atomic Energy - 1968," U.S. Atomic Energy Commission, NTIS #TID-24190, 1968.

Spicer, T. O. and J. A. Havens, "Development of Vapor Dispersion Models for Non-Neutrally Buoyant Gas Mixtures--Analysis of USAF/N_2O_4 Test Data," USAF Engineering and Services Laboratory, ESL-TR-86-24, Final Report, September, 1986.

Spicer, T. O. and J. A. Havens, "Modeling HF and NH_3 Spill Test Data using DEGADIS," Paper No. 87b, 1988 Summer National Meeting of American Institute of Chemical Engineers, August, 1988.

Spicer, T. O. and J. A. Havens, "Development of Vapor Dispersion Models for Nonneutrally Buoyant Gas Mixtures--Analysis of TFI/NH_3 Test Data," UASF Engineering and Services Laboratory, Final Report, October, 1988.

Treybal, R. E., Mass Transfer Operations, 3rd edition, McGraw-Hill, New York, 1980.

Turner, D. B., "Workbook of Atmospheric Dispersion Estimates," USEPA 999-AP-26, U.S. Environmental Protection Agency, Washington DC, 1970.

van Ulden, A. P., "The Unsteady Gravity Spread of a Dense Cloud in a Calm Environment," 10th International Technical Meeting on Air Pollution Modeling and its Applications, NATO-CCMS, Rome, Italy, October, 1979.

van Ulden, A. P., "A New Bulk Model for Dense Gas Dispersion: Two-Dimensional Spread in Still Air," I.U.T.A.M. Symposium on Atmospheric Dispersion of Heavy Gases and Small Particles, Delft University of Technology, The Netherlands, August 29-September 2, 1983.

Wheatley, C. J., "Discharge of Ammonia to Moist Atmospheres--Survey of Experimental Data and Model for Estimating Initial Conditions for Dispersion Calculations," UK Atomic Energy Authority Report SRD R410, 1986.

Zeman, O. and H. Tennekes, "Parameterization of the Turbulent Energy Budget at the Top of the Daytime Atmospheric Boundary Layer," Journal of the Atmospheric Sciences, January, 1977.

(This page is intentionally left blank.)

APPENDIX A

MODEL APPLICATION ON VAX/VMS

DEGADIS Version 2.1 (including the jet/plume front-end model) was developed under VAX/VMS V3.5 and VAX-11 Fortran V3.5; there should be no installation difficulty for VAX/VMS V4.0 or later.

Installation

The directory which contains the Fortran source code for DEGADIS must be equivalenced with the logical name SYS$DEGADIS:. If the full directory specification is DQA0:[DEGADIS_V21], issue the VAX/VMS command:

```
$ ASSIGN DQA0:[DEGADIS_V21] SYS$DEGADIS:
```

with either the /PROCESS, /GROUP, or /SYSTEM qualifier (/SYSTEM is recommended). Once this assignment is made, the files must be compiled and linked to form DEGADISIN, DEGADIS1, DEGADIS2, DEGADIS3, DEGADIS4, and SDEGADIS2; the VMS command procedure BUILD.COM performs these tasks. The process which compiles and links DEGADIS must have READ, WRITE, and EXECUTE access privileges to SYS$DEGADIS while only READ and EXECUTE access privileges are needed to execute the existing models.

Considerations for Installation other than VAX/VMS

There are two types of problems which may occur when attempting to install DEGADIS on a different computer or operating system. The first is due to the use of non-standard ANSI Fortran 77 language elements. The second is due to the use of external VAX/VMS routines in DEGADIS.

The following VAX-11 Fortran extensions have been used in DEGADIS:

(*) In-line comments--An exclamation mark (!) is used to include comments at the end of a valid statement.

(*) Special characters--The underscore (_) is used in variable names.

(*) DO loops--DO loops are used with the structure:

```
DO v = e1,e2[,e3]
  .
  .
  .
END DO
```

where v is a variable name and e1, e2, and e3 are numeric expressions. The numeric expressions have the standard Fortran 77 meaning.

(*) INCLUDE statements--INCLUDE statements simply allow other source files to be inserted in the routine being compiled at this point in the source. The system table '($SSDEF)' is used to check the status or returning system routines.

(*) OPEN keyword NAME--The OPEN keyword NAME specifies the file name to be opened.

(*) Fortran descriptors--The Q descriptor obtains the integer number of characters remaining in the input record during a READ operation. The dollar sign ($) descriptor suppresses the carriage return at the end of a line on output.

(*) Continuation lines--Continuation lines have been expressed by using either a nonblank character in column 6 or by beginning the line with a tab and a number in the next column.

(*) Concatenation of character strings--Character strings are concatenated using two slashes (//).

The following VAX/VMS subroutines have been used in DEGADIS:

(*) SECNDS
TIME = SECNDS(TIME0)
SECNDS returns to TIME the difference between the number of seconds after midnight on the system clock and the value of TIME0.

(*) LIB$DATE_TIME
ISTAT = LIB$DATE_TIME (STRING)
LIB$DATE_TIME returns a 24-character ASCII string with the system date and time. ISTAT is an integer variable which accepts the return status.

(*) LIB$DO_COMMAND
ISTAT = LIB$DO_COMMAND (STRING)
LIB$DO_COMMAND issues the command STRING (a character string) to VAX/VMS. If the command is not issued, ISTAT contains the failure code. If the command is issued, the calling process is terminated.

Model Implementation

The (combined) jet/plume (JETPLU) and DEGADIS models consist of nine separate programs:

• JETPLU_IN	is the file-based input module which defines the simulation.
• JETPLU	describes the trajectory and dilution of the vertically oriented jet release.
• DEGBRIDGE	is an interface program which takes the output from JETPLU (if where the jet/plume returns to ground level) and provides appropriate input to DEGADIS.

DEGADISIN	is the interactive input module which defines DEGADIS-only simulation.
DEGADIS1	determines α and describes the gas source for transient and steady-state releases.
DEGADIS2	describes the pseudosteady-state downwind dispersion of the released gas.
DEGADIS3	sorts the results of DEGADIS2 for a transient release at given times.
DEGADIS4	sorts the results of DEGADIS2 for a transient release at given positions.
SDEGADIS2	describes the steady-state downwind dispersion of the released gas.

Steady-state releases are simulated by executing JETPLU, DEGBRIDGE, DEGADIS1, and SDEGADIS2; time-limited (transient) releases are simulated by executing JETPLU, DEGBRIDGE, DEGADIS1, DEGADIS2, and DEGADIS3.

The JETPLU DEGADIS model uses three types of input information
- VAX/VMS command procedure for execution
- simulation definition
- numerical parameters.

The VAX/VMS command procedure used to execute JETPLU DEGADIS is generated by JETPLU_IN which also reads the simulation definition. DEGADIS can be run without JETPLU (for ground-level, zero momentum releases). Input to DEGADIS is made to run interactively using DEGADISIN. Example input sessions are included in Appendix B. The numerical parameters (convergence criteria, initial increments, etc.) are supplied to DEGADIS through a series of input files. Although these numerical parameters are easily changed, the user should need to change these only rarely (with the exception of the time-sort parameters which are explained later in this appendix).

VAX/VMS Command Procedures

The VAX/VMS command procedure generated by JETPLU_IN controls the execution of programs for the simulation. Program execution follows one of two paths: (a) a transient (time-limited) release or (b) a steady-state release. JETPLU_IN, which automatically generates the appropriate command procedure, requires a simulation name be specified. The simulation name must be a valid VAX/VMS file name without a file extension and is designated RUN_NAME. JETPLU and DEGADIS will use this file name with standard extensions for input, interprocess communication, and output. Figures A.1 and A.2 show example VAX/VMS command procedures created by JETPLU_IN for the run names TEST_S and TEST for steady-state and transient releases, respectively. The directory which contains the executable images has been assigned the system logical name SYS$DEGADIS. The COPY/LOG command copies a file from the first argument to the second argument, and the RUN command executes the specified image. These steps may also be carried out by issuing the commands at a terminal.

```
$ assign test_s.ino for001
$ assign test_s.out for003
$ assign test_s.ind for002
$ run sys$degadis:jetplu
$ deassign for001
$ deassign for002
$ deassign for003
$ run sys$degadis:degbridge
test_s
$ copy/log SYS$DEGADIS:example.er1 test_s.er1
$ copy/log SYS$DEGADIS:example.er2 test_s.er2
$ run SYS$DEGADIS:DEGADIS1
test_s
$ run SYS$DEGADIS:SDEGADIS2
test_s
$ copy/log test_s.out + test_s.scl + test_s.sr3 -
  test_s.lis
```

Figure A.1. Example DEGADIS Command Procedure on VAX/VMS for a Steady-State Simulation Named TEST_S.

```
$ assign test.ino for001
$ assign test.out for003
$ assign test.ind for002
$ run sys$degadis:jetplu
$ deassign for001
$ deassign for002
$ deassign for003
$ run sys$degadis:degbridge
test
$ copy/log SYS$DEGADIS:example.er1 test.er1
$ copy/log SYS$DEGADIS:example.er2 test.er2
$ copy/log SYS$DEGADIS:example.er3 test.er3
$ run SYS$DEGADIS:DEGADIS1
test
$ run SYS$DEGADIS:DEGADIS2
test
$ run SYS$DEGADIS:DEGADIS3
TEST
$ copy/log test.out + test.scl + test.sr3 -
  test.lis
```

Figure A.2. Example DEGADIS Command Procedure on VAX/VMS for a Transient Simulation Named Test.

Simulation Definition

JETPLU_IN specifies information about the ambient wind field, the properties of the released material, and details of the release.

The input uses free-formatted files; a value for each parameter must be specified in the input file even if it has no meaning for the simulation at hand. This form of input allows the user to ignore spacing in the input file. Numbers on a particular line can be separated by commas, spaces, or tabs. Comments may also be included at the end of any line in the input file as long as <u>all</u> required values are given before <u>any</u> comments are included. Lines which are only comments are allowed only after all required values have been specified. (Sample input files are included in Appendix B.)

The ambient wind field is characterized by a known velocity u_0 at a given height z_0, a surface roughness z_R, and the Pasquill stability class or Monin-Obukhov length. The Pasquill stability class is required to estimate values of the lateral, vertical, and along-wind similarity coefficients. If specified, the Monin-Obukhov length λ is required to calculate the friction velocity u_*. In addition to these specifications, the ambient temperature, pressure, and humidity must be specified.

The contaminant properties are used by the model to fix the mixture density from the contaminant concentration. In the model, there are three possible ways to simulate the mixture density:

Case 1: Treat the contaminant as an ideal gas and allow for condensation of ambient humidity. This is accomplished by setting <NDEN>=0. In DEGADIS, ground-to-cloud heat transfer can either by included (<INDHT>=1) or precluded (<INDHT>=0). The contaminant heat capacity is determined by <CPK> and <CPP> using

$$c_{p_c}(T) = (MW_c)^{-1}\left[3.33 \times 10^4 + q_1\left[\frac{T^{p_1} - T_0^{p_1}}{T - T_0}\right]\right] \tag{A.1}$$

where <CPK> is denoted by q_1 and <CPP> is denoted by p_1. If a constant heat capacity is desired, set <CPP> to 0. and <CPK> to the desired heat capacity (in J/kg K).

Case 2: Treat the contaminant as an ideal gas and do not allow for condensation of ambient humidity. In this case, ground-to-cloud heat transfer in DEGADIS is not allowed, and $(\rho - \rho_a)/c$ is assumed constant. This is accomplished by setting <NDEN>=-1. Values input for <INDHT>, <CPK>, and <CPP> are ignored.

Case 3: Estimate the contaminant/air mixture density as a function of contaminant concentration based on a user-supplied estimate. Ordered triples of the contaminant mole fraction, the contaminant concentration (kg/m^3), and the contaminant/air mixture density (kg/m^3) are input to the model. This is accomplished by setting <NDEN> equal to the number of triples. In the input file, the ordered triples (starting with pure air) are input on the next NDEN lines immediately following the line for <NDEN>. Values input for <INDHT>, <CPK>, <CPP>, and <TEMJET> are ignored.

The user must choose to simulate the release as time-limited (transient) or steady-state (by the specification of <TEND>), and the jet elevation and diameter must be specified.

Figure A.3 summarizes the structure of the input file RUN_NAME.IN required by JETPLU_IN. The description following is written for each of the variables in Figure A.3. After each line of input, an explanation is included. All units are SI (meters, kilograms, second) except as specified.

```
<TITLE1>
<TITLE2>
<TITLE3>
<TITLE4>
<U0>   <Z0>
<ZR>
<INDVEL> <ISTAB> <RML>
<TAMB> <PAMB> <RELHUM>
<TSURF>
<GASNAM>
<GASMW>
<AVTIME>
<TEMJET>
<GASUL> <GASLL> <ZLL>
<INDHT> <CPK> <CPP>
<NDEN>
<ERATE>
<ELEJET> <DIAJET>
<TEND>
<DISTMX>
```

Figure A.3. RUN_NAME.IN Structure Required by JETPLU_IN

<TITLE1>
<TITLE2>
<TITLE3>
<TITLE4>

<TITLE1>, <TITLE2>, <TITLE3>, and <TITLE4> are four lines of up to 80 characters each of a title for this simulation.

<U0> <Z0>

<U0> (m/s) is the ambient wind velocity at <Z0> (m).

<ZR>

<ZR> is the surface roughness (m).

<INDVEL> <ISTAB> <RML>

<INDVEL> is an indicator which determines the method of calculation for the ambient velocity profile in the jet/plume model as follows:

For <INDVEL>=1, the Pasquill-Gifford stability category (in <ISTAB> using 1 for A, 2 for B, etc.) is used along with

<ZR> to determine the Monin-Obukhov length <RML>; the log velocity profile is then determined using <RML>.

For <INDVEL>=2, the Monin-Obukhov length <RML> is supplied by the user; the log velocity profile is then determined using <RML>. Note that <ISTAB> must still be specified.

<TAMB> <PAMB> <RELHUM>

<TAMB>, <PAMB>, and <RELHUM> are the ambient temperature (K), the ambient pressure (atm or N/m^2), and the relative humidity (%), respectively.

<TSURF>

<TSURF> is the surface temperature (K); if <TSURF> < 250 K, <TSURF> is set to <TAMB>.

<GASNAM>

<GASNAM> is a three-letter designation for the contaminant name. Any character string of three letters or less is valid; this is for user run identification and does not access property data.

<GASMW>

<GASMW> is the contaminant molecular weight (kg/kmole).

<AVTIME>

<AVTIME> is the averaging time (s). This parameter is used to estimate the value of <DELTAY>.

<TEMJET>

<TEMJET> is the temperature of the released contaminant (K).

<GASUL> <GASLL> <ZLL>

<GASUL> and <GASLL> are the concentrations to be used for estimating contours for an upper and lower concentration level in DEGADIS. The calculations are made for the elevation <ZLL>. Note that the JETPLU/DEGADIS computations will be carried out to <GASLL>/2.

<INDHT> <CPK> <CPP>

<INDHT> is used to include heat transfer in the DEGADIS computations. Heat transfer is not included for <INDHT>=0. For <INDHT>=1, heat transfer is included, and the heat transfer coefficient is calculated by DEGADIS. <CPP> and <CPK> are used to calculate the heat capacity as a function of temperature according to the correlation included in DEGADIS. If a constant heat capacity is desired, set <CPP> to 0. and <CPK> to the desired heat capacity (J/kg K).

<NDEN>

<NDEN> is used to specify the contaminant density profile. There are three cases for <NDEN>:

<NDEN> = -1; The simulation treats the contaminant as if it were an ideal gas with a molal heat capacity equal to that of air. Water condensation effects are ignored (equivalent to an "isothermal" release in DEGADIS).

<NDEN> = 0; The simulation treats the contaminant as if it were an ideal gas with the heat capacity indicated by <CPK> and <CPP>. Water condensation effects are taken into account as appropriate (equivalent to a nonisothermal release in DEGADIS).

<NDEN> > 0; <NDEN> specifies the number of triples which follow in the next <NDEN> lines. The triples are used to specify the contaminant concentration as a function of density based on adiabatic mixing with ambient air. The ordered triples represent (in order):

(1) the contaminant mole fraction
(2) the contaminant concentration (kg contam/m^3 mix)
(3) the mixture density (kg mixture/m^3 mixture).

The ordered triples must go from pure air to pure contaminant (equivalent to an "isothermal" release in DEGADIS).

<ERATE>

<ERATE> is the mass evolution (release) rate (kg/s).

<ELEJET> <DIAJET>

<ELEJET> is the initial jet elevation (m), and <DIAJET> is the initial jet diameter (m).

<TEND>

<TEND> is the duration of the primary release (s). For steady-state releases, set <TEND> to 0.; to run the jet/plume model only, set <TEND> to a negative number.

<DISTMX>

<DISTMX> is the maximum distance between output points in the JETPLU output (m).

DEGBRIDGE takes the output from the jet/plume model and creates the file necessary for DEGADIS to complete the simulation.

Input Module--DEGADISIN

DEGADISIN is the optional interactive input module which defines a DEGADIS simulation; DEGADISIN is composed of two subroutines (Figure A.4):

(*) DEGADISIN contains the program overhead and generates the command file RUN_NAME.COM which can be used to control simulation execution (D-38).

(*) IOT contains the interactive question-and-answer sequence which defines the simulation; IOT also creates the file RUN_NAME.INP (D-76).

An example of a DEGADISIN query sequence is included in Appendix B. As this information is gathered, it is written to the file RUN_NAME.INP (Figure A.5). Once DEGADISIN is completed, RUN_NAME.INP may be edited to correct minor input mistakes. If major revisions are necessary, the recommended practice is to execute DEGADISIN again.

Once the information required by DEGADISIN has been entered properly, DEGADIS may be executed using the command procedure generated by DEGADISIN under the file name RUN_NAME.COM. If DEGADIS is not to be run using this command file, the user must enter the simulation name (RUN_NAME) after each of the programs are begun. (In this case, the user must also provide copies of the numerical parameter files.)

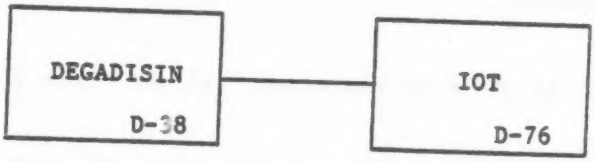


Figure A.4. DEGADISIN Flowchart.

```
                              TITLE(1)
                              TITLE(2)
                              TITLE(3)
                              TITLE(4)
                              UO, ZO, ZR
                              ISTAB
                              OODIST,AVTIME
                              DELTA, BETA, RML
                              SIGX_COEFF, SIGX_POW, SIGX_MIN_DIST
                              TAMB, PAMB, HUMID
                              ISOFL, TSURF
                              IHTFL, HTCO
                              IWTFL, WTCO
                              GAS_NAME
                              GAS_MW, GAS_TEMP, GAS_RHOE
                              GAS_CPK, GAS_CPP
                              GAS_UFL, GAS_LFL, GAS_ZSP
                           {  NP
If <ISOFL≠0) then          {  DEN((J,1),J=1,5)
(for                       {  DEN((J,2),J=1,5)
external density           {      .
calculations)              {      .
                           {      .
                           {  DEN((J,NP),J=1,5
                              CCLOW
                              GMASS0
                              NT
                  {  PTIME(1),ET(1),R1T(1),PWC(1),PTEMP(1),PFRACV(1)
                  {  PTIME(2),ET(2),R1T(2),PWC(2),PTEMP(2),PFRACV(2)
NT of these       {      .
                  {      .
                  {      .
                  {  PTIME(NT),ET(NT),R1T(NT),PWC(NT),PTEMP(NT),PFRACV(NT)
                     CHECK1, CHECK2, AGAIN, CHECK3, CHECK4, CHECK5
                     TINP
for steady-state     ESS, SRCLEN, SRCB
  only
```

Figure A.5. Structure for Free-Formatted RUN_NAME.INP File.

Source Module--DEGADIS1

DEGADIS1 estimates values for the friction velocity and ambient wind profile power α and characterizes the primary gas source for the remainder of the model; DEGADIS1 is composed of the following subroutines (Figure A.6):

(*) AFGEN	linearly interpolates between a pair of points based on a list of supplied values (D-3).

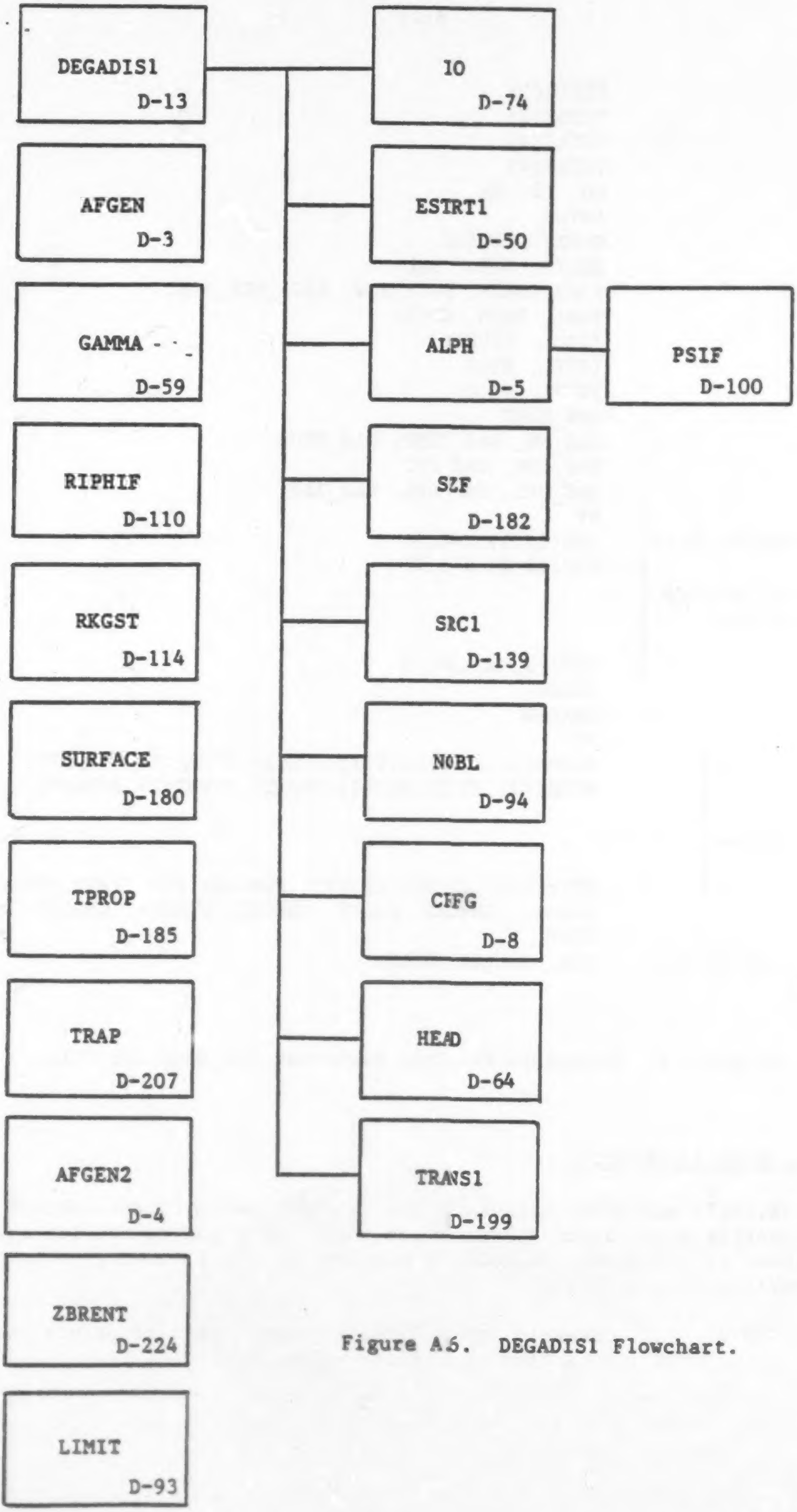

DEGADIS1
D-13
AFGEN
D-3
GAMMA
D-59
RIPHIF
D-110
RKGST
D-114
SURFACE
D-180
TPROP
D-185
TRAP
D-207
AFGEN2
D-4
ZBRENT
D-224
LIMIT
D-93
IO
D-74
ESTRT1
D-50
ALPH
D-5
PSIF
D-100
SZF
D-182
SRC1
D-139
NOBL
D-94
CIFG
D-8
HEAD
D-64
TRANS1
D-199

Figure A6. DEGADIS1 Flowchart.

(*) AFGEN2	linearly interpolates between a pair of points based on a list of supplied values (D-4).
(*) ALPH	estimates the ambient wind profile power α by minimizing the integral of the difference between an ambient logarithmic velocity profile and the assumed power law velocity profile (D-5).
(*) CRFG	creates a table of calculated values which will describe the secondary gas source for the downwind dispersion calculations (D-8).
(*) DEGADIS1	contains the program overhead and sequentially calls the routines required to estimate the ambient wind profile power α and to characterize the primary gas source (D-14).
(*) ESTRT1	recovers the numerical parameters contained in the file RUN_NAME.ER1 (D-50).
(*) GAMMA	calculates the gamma function of the argument (I.e. $\Gamma(x)$) (D-59).
(*) HEAD	writes a formatted output heading to the file RUN_NAME.SCL (D-64).
(*) IO	recovers the simulation definition contained in RUN_NAME.INP (D-74).
(*) LIMIT	establishes the limit for ZBRENT (D-93).
(*) NOBL	estimates gas source behavior when no gas blanket is present (D-94).
(*) PSIF	calculates the ψ function in the logarithmic velocity profile (D-100).
(*) RIPHIF	calculates the Richardson number and the values of $\phi(Ri_*)$ and $\hat{\phi}(Ri_*)$ (D-110).
(*) RKGST	performs numerical integration of a specified system of equations using a variable-step, modified fourth-order Runge-Kutta method (D-114).
(*) SCR1	contains the ordinary differential equations which describe the gas blanket formed as a result of the primary gas source (D-139).

(*) SURFACE	estimates heat and water transfer rates across the bottom surface of the gas layer (D-180).
(*) SZF	estimates the value of S_z if the primary source can just form a gas blanket over the source (D-182).
(*) TPROP	is a series of utility routines which estimate the thermodynamic properties of a given gas mixture (D-185).
(*) TRANS1	writes the information to continue the next simulation step to the file RUN_NAME.TR2 (D-199).
(*) TRAP	is a utility included for program diagnostics (D-207).
(*) ZBRENT	finds the bracketed root of an equation using Brent's method (D-224).

As input, DEGADIS1 requires two files:

(*) RUN_NAME.ER1	contains various numerical parameters. For most simulations, a copy of the SYS$DEGADIS:EXAMPLE.ER1 file will be adequate. A copy of SYS$DEGADIS:EXAMPLE.ER1 is included in Figure A.7.
(*) RUN_NAME.INP	contains the simulation definition as discussed in Appendix B (Figure A.5).

As output, DEGADIS1 generates the following files:

(*) RUN_NAME.SCD	contains the calculated values which describe the secondary gas source. It is generated by SRC1 and NOBL and is then read by CRFG; it is a temporary file.
(*) RUN_NAME.SCL	is the listed output which describes the input information for the simulation and the calculated secondary gas source. It is written by HEAD and CRFG.
(*) RUN_NAME.TR2	contains the information to continue the next simulation step.

```
!This is an example of how to set up and use the run parameter
! input files.  Comment lines start with an exclamation mark(!)
! in the first column.  The only restrictions for data input are
! as follows:
!       1) The data must be entered in the same order
!          all of the time.
!       2) Only the number must be between columns 10 and 20.
!       3) Always include the decimal point in the number
!
! Column layout:
!23456789012345678901234567890
!--------1---------2---------3
!        I         I
STPIN      0.01        MAIN - RKGST - INITIAL STEP SIZE
ERBND      0.0025      MAIN - RKGST - ERROR BOUND
STPMX      5.12        MAIN - RKGST - MAXIMUM STEP SIZE
WTRG        1.         MAIN - RKGST - WEIGHT FOR RG
WTTM        1.         MAIN - RKGST - WEIGHT FOR Total Mass
WTYA        1.         MAIN - RKGST - WEIGHT FOR ya
WTYC        1.         MAIN - RKGST - WEIGHT FOR yc
WTEB        1.         MAIN - RKGST - WEIGHT FOR Energy Balance
WTmB        1.         MAIN - RKGST - WEIGHT FOR Momentum Balance
WTuh        1.         MAIN - RKGST - WEIGHT FOR Ueff*Heff
XLI        0.05        ALPH - LOWER LIMIT OF SEARCH FOR ALPHA
XRI        0.50        ALPH - UPPER LIMIT OF SEARCH FOR ALPHA
EPS        0.001       ALPH - ERROR BOUND USED BY "RTMI"
ZLOW       0.01        ALPHI - maximum BOTTOM HEIGHT FOR FIT OF ALPHA
!
STPINZ     -0.02       ALPHI - INITIAL RKGST STEP <0.
!
ERBNDZ     0.005       ALPHI - ERROR BOUND FOR RKGST
!
STPMXZ     -2.00       ALPHI - MAXIMUM STEP FOR RKGST <0.
!
!  Note that comment lines can be mixed with the numbers.
!
SRCOER     0.007       SRC1O - OUTPUT Error criterion
SRCSS       5.2        SRC1O - min time for Steady: STPMX
SRCcut     .00001      SRC1O - min height for blanket
htcut      .0          SRC1 - min height for blanket heat transfer
ERNOBL     1.0005      NOBL - CONVERGENCE ratio
NOBLPT     100.        NOBL - NUMBER OF POINTS
!                            USED ON THE LAST PORTION OF THE SOURCE
!
crfger     0.008       error criterion in building GEN3 vectors
!
```

Figure A.7. SYS$DEGADIS:EXAMPLE.ER1 Listing.

```
epsilon     0.59      epsilon USED IN AIR ENTRAINMENT SPECIFICATION
!
! /SPRD_CON/
!
ce          1.15      constant in gravity slumping equation
delrhomin   0.0       stop cloud spread if delrho<delrhomin
!
!
! /SZFC/
!
szstp0      0.01      SZF - Initial step size
szerr       0.001     SZF - Error criterion
szstpmx     5.0       SZF - Maximum step size [-] m
szsz0       0.01      SZF - Initial Value of dellay*Ueff*Heff
!
!
!    /ALPHcom/
!
ialpfl      1.        ALPHI - calculation flag; 0) alpha=alpco; 1)1/(1+z); 2)1
alpco       0.2       ALPHI - Value for alpha if IALPFL = 0
!
!
!    /PHIcom/
!
iphifl      3.        PHIF - calc flag
dellay      2.15      Ratio of Hl/Heff
!
!
!
!    /VUcom/
!
vua         1.3       Constant Av in source model
vub         1.2       Constant Bv in source model
vuc         20.0      Constant Ev in source model
vud         .64       Constant Dv in source model
vudelta     0.20      Constant DELTAv in source model
!
! End-of-File
```

Figure A.7. (concluded)

Pseudosteady-State Module--DEGADIS2

DEGADIS2 performs the downwind dispersion portion of the calculation for each of several observers released successively over the transient source described by DEGADIS1. (Note that the routines INCGAMMA, GAMMA, and SERIES are linked in DEGADIS2 but never are called in DEGADIS2.) DEGADIS2 is composed of the following subroutines (Figure A.8):

(*) AFGEN	linearly interpolates between a pair of points based on a list of supplied values (D-3).
(*) AFGEN2	linearly interpolates between a pair of points based on a list of supplied values (D-4).
(*) DEGADIS2	contains the program overhead and sequentially calls the routines to recover the information generated in DEGADIS1, recover the numerical parameter file RUN_NAME.ER2, and perform the simulation (D-23).
(*) ESTRT2	recovers the numerical parameters contained in the file RUN_NAME.ER2, particularly the number of observers NOBS (D-54).
(*) LIMIT	establishes the limit for ZBRENT (D-93).
(*) OB	contains the ordinary differential equations which average the gas source for each observer (D-97).
(*) PSS	contains the ordinary differential equations which describe the portion of the downwind dispersion calculation when $b > 0$ (D-100).
(*) PSSOUT	governs the output of calculated points to the file RUN_NAME.PSD when PSS is active (D-104).
(*) RIPHIF	is a series of utilities which calculates the Richardson number and the value of $\phi(Ri)$ (D-110).
(*) RKGST	performs numerical integration of a specified system of equations using a modified fourth-order Runge-Kutta method (D-114).
(*) SSG	contains the ordinary differential equations which describe the portion of the downwind dispersion calculation when $b = 0$ (D-152).
(*) SSGOUT	governs the output of calculated points to the file RUN_NAME.PSD when SSG is active (D-155).

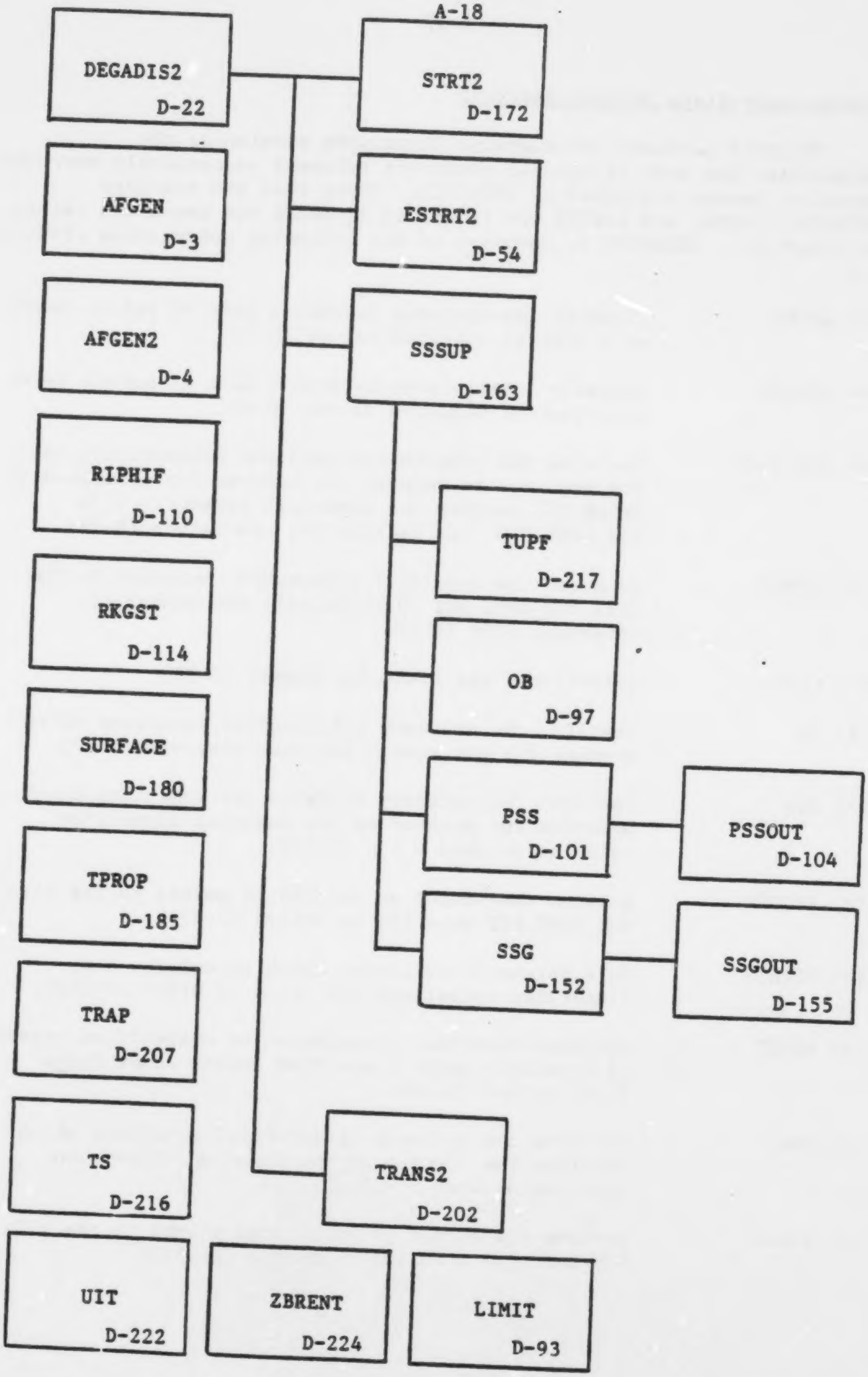

DEGADIS2
D-22
STRT2
D-172
AFGEN
D-3
ESTRT2
D-54
AFGEN2
D-4
SSSUP
D-163
RIPHIF
D-110
TUPF
D-217
RKGST
D-114
OB
D-97
SURFACE
D-180
PSS
D-101
PSSOUT
D-104
TPROP
D-185
SSG
D-152
SSGOUT
D-155
TRAP
D-207
TS
D-216
TRANS2
D-202
UIT
D-222
ZBRENT
D-224
LIMIT
D-93

Figure A.8. DEGADIS2 Flowchart.

(*) SSSUP	is a supervisor routine which controls the averaging of the source for each observer, the portion of the downwind dispersion calculation when b > 0, and the portion of the downwind dispersion calculation when b = 0 (D-163).
(*) STRT2	recovers the information generated in DEGADIS1 contained in the file RUN_NAME.TR2 (D-172).
(*) SURFACE	estimates heat and water transfer rates across the bottom surface of the gas layer (D-180).
(*) TPROP	is a series of utility routines which estimates the thermodynamic properties of a given gas mixture (D-185).
(*) TRANS2	writes the information necessary for DEGADIS3 to the file (RUN_NAME.TR3) (D-202).
(*) TRAP	is a utility included for program diagnostics (D-207).
(*) TS	calculates the time when a given observer will be at a given downwind distance (D-216).
(*) TUPF	contains the routines which determine the intersection of the upwind/downwind edge of the secondary gas source with a given observer (D-217).
(*) UIT	is a series of routines to calculate observer position and velocity as a function of time (D-222).
(*) ZBRENT	finds the bracketed root of an equation using Brent's method (D-224).

As input, DEGADIS2 requires two files:

(*) RUN_NAME.ER2	contains the various numerical parameters, particularly the number of observers NOBS. For most simulations, a copy of the SYS$DEGADIS:EXAMPLE.ER2 file will be adequate. A copy of SYS$DEGADIS:EXAMPLE.ER2 is included in Figure A.9.
(*) RUN_NAME.TR2	contains the basic simulation definition as well as calculated secondary source parameters.

DEGADIS2 generates the following output files:

(*) RUN_NAME.OBS	contains a summary of the source parameters for each observer.

```
! This is an example for an "ER2" run parameter file.
! The same rules apply as for the "ER1" files.
!
!234567890123456789012345678901234567890
!--------1---------2---------3
!
! These values are in common area /ERROR/
!
SY0ER       0.0       SSSUP - RKGST - INITIAL SY
ERRO        0.005     SSSUP - RKGST(OBS) - ERROR BOUND
SZ0ER       0.01      SSSUP - RKGST(OBS) - INITIAL SZ
WTAIO       1.0       SSSUP - RKGST(OBS) - WEIGHT FOR AI
WTQOO       1.0       SSSUP - RKGST(OBS) - WEIGHT FOR Q
WTSZO       1.0       SSSUP - RKGST(OBS) - WEIGHT FOR SZ
ERRP        0.003     SSSUP - RKGST(PSS) - ERROR BOUND
SMXP        10.       SSSUP - RKGST(PSS) - MAXIMUM STEP
WTSZP       1.0       SSSUP - RKGST(PSS) - WEIGHT FOR SZ
WTSYP       1.0       SSSUP - RKGST(PSS) - WEIGHT FOR SY
WTBEP       1.0       SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
WTDH        1.0       SSSUP - RKGST(PSS) - WEIGHT FOR DH
ERRG        0.003     SSSUP - RKGST(SSG) - ERROR BOUND
SMXG        10.       SSSUP - RKGST(SSG) - MAXIMUM STEP SIZE
ERTDNF      0.0005    TDNF - CONVERGENCE CRITERION
ERTUPF      0.0005    TUPF - CONVERGENCE CRITERION
WTRUH       1.0       SSSUP - RKGST(SSG) - WEIGHT FOR RUH
WTDHG       1.0       SSSUP - RKGST(SSG) - WEIGHT FOR DH
!
! These values are in common area /STP/
!
STPO        0.05      SSSUP - RKGST(OBS) - INITIAL STEP
STPP        0.05      SSSUP - RKGST(PSS) - INITIAL STEP
ODLP        0.06      SSSUP - RKGST(PSS) - RELATIVE OUTPUT DELTA
ODLLP       80.       SSSUP-RKGST(PSS)-MAXIMUM DISTANCE BETWEEN
OUTPUTS(m)
STPG        0.05      SSSUP - RKGST(SSG) - INITIAL STEP
ODLG        0.045     SSSUP - RKGST(SSG) - RELATIVE OUTPUT DELTA
ODLLG       80.       SSSUP-RKGST(SSG)-MAXIMUM DISTANCE BETWEEN
OUTPUTS(m)
!
! The last variable NOBS is in /CNOBS/
!
! Note: it is read in as a real value even though it is integer type
!     in the program.
!
NOBS        30.
!
!
!
! End-of-File
```

Figure A.9. SYS$DEGADIS:EXAMPLE.ER2 Listing.

(*) RUN_NAME.PSD	contains the calculated downwind dispersion parameters for each observer. DEGADIS3 and DEGADIS4 sort this information to determine the downwind concentration profiles as a function of position and time.
(*) RUN_NAME.TR3	contains the simulation definition and the number of each record type written to RUN_NAME.PSD.

Time Sort Modules--DEGADIS3 and DEGADIS4

DEGADIS3 sorts the downwind dispersion calculation for each of several observers for concentration information at several given times; the along-wind dispersion correction is then applied as desired. DEGADIS3 uses the following subroutines (Figure A.10).

(*) DEGADIS3	contains the program overhead and sequentially calls the routines to recover the information generated in DEGADIS2, recover the numerical parameter file RUN_NAME.ER3, sort and apply the along-wind dispersion correction to the results of DEGADIS2, and output the results (D-28).
(*) ESTRT3	recovers the numerical parameters contained in the file RUN_NAME.ER3, particularly the time sort parameters (D-58).
(*) GAMMA	calculates the gamma function of the argument (i.e. $\Gamma(x)$) D-59).
(*) GETTIM	sets the default time sort parameters as needed (D-61).
(*) INCGAMMA	calculates the incomplete gamma function of the two arguments (D-70).
(*) LIMIT	establishes the limit for ZBRENT (D-93).
(*) SERIES	evaluates a series needed to estimate the mass of gas above a given concentration level (D-130).
(*) SORTS	recovers the information in RUN_NAME.PSD and arranges the information according to the time sort parameters in the file RUN_NAME.ER3 (D-131).
(*) SORTS1	applies the along-wind dispersion correction to the time-sorted information (D-134).
(*) SRTOUT	generates the formatted output file RUN_NAME.SR3 (D-148).
(*) STRT3	recovers the information generated in DEGADIS2 contained in the file RUN_NAME.TR3 (D-178).

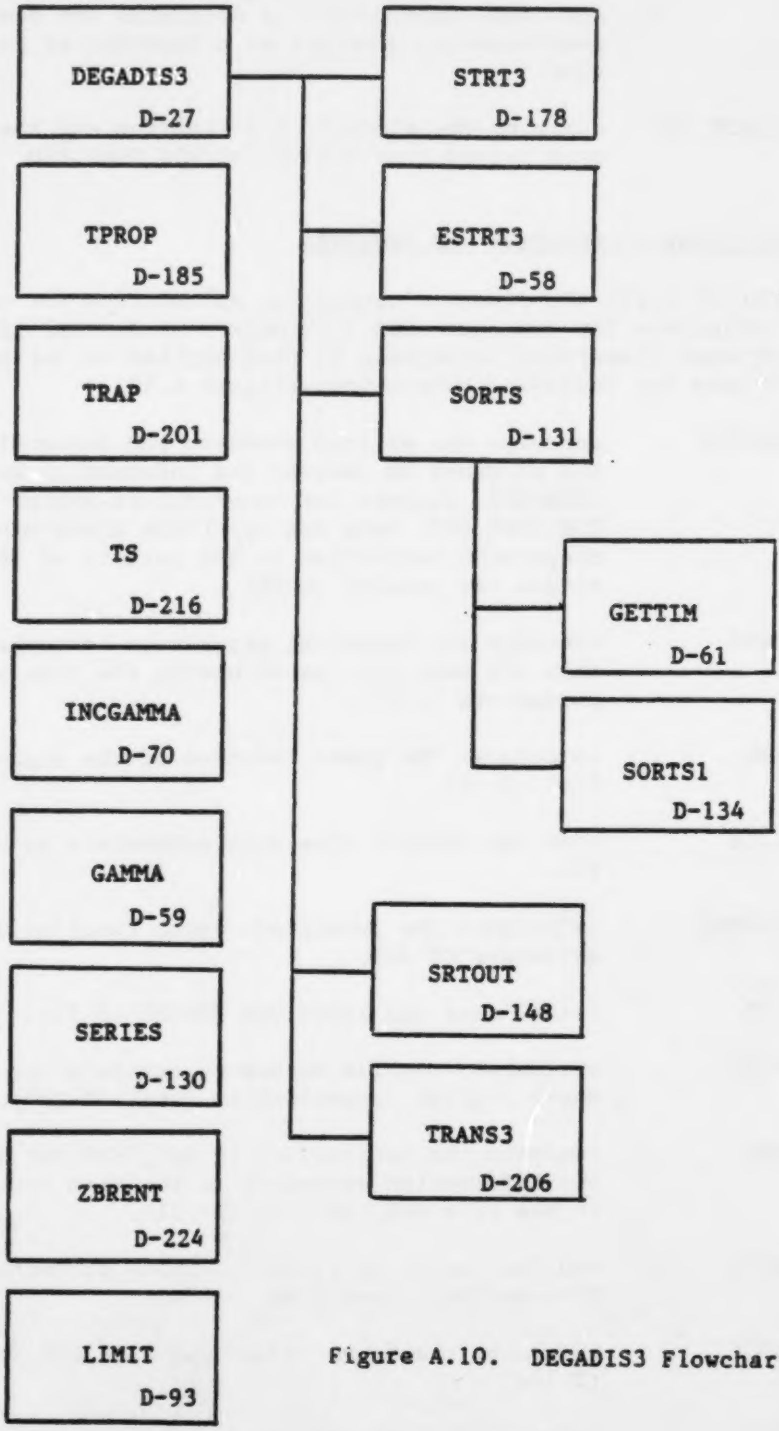

DEGADIS3
D-27
TPROP
D-185
TRAP
D-201
TS
D-216
INCGAMMA
D-70
GAMMA
D-59
SERIES
D-130
ZBRENT
D-224
LIMIT
D-93
STRT3
D-178
ESTRT3
D-58
SORTS
D-131
GETTIM
D-61
SORTS1
D-134
SRTOUT
D-148
TRANS3
D-206

Figure A.10. DEGADIS3 Flowchart.

(*) TPROP	is a series of utility routines which estimate the thermodynamic properties of a given gas mixture (D-894).
(*) TRANS3	writes RUN_NAME.TR4 which contains the necessary information to recover the other output files for this simulation (D-206).
(*) TRAP	is a utility included for program diagnostics (D-207).
(*) TS	calculates the time when a given observer will be at a given downwind distance (D-216).
(*) ZBRENT	finds the bracketed root of an equation using Brent's method (D-224).

As input, DEGADIS3 requires three files:

(*) RUN_NAME.ER3	contains various numerical parameters including the time sort parameters and the flag which dictates whether the along-wind dispersion correction is applied. A copy of SYS$DEGADIS:EXAMPLE.ER3 file uses the default time sort parameters and includes the along-wind dispersion correction which should apply for most simulations. A copy of SYS$DEGADIS:EXAMPLE.ER3 is included in Figure A.11.
(*) RUN_NAME.PSD	contains the calculated downwind dispersion parameters for each observer. DEGADIS3 sorts this information to determine the downwind concentration profiles as a function of position at a given time.
(*) RUN_NAME.TR3	contains the number of each record type written to RUN_NAME.PSD as well as the simulation definition.

As output, DEGADIS3 generates two new files:

(*) RUN_NAME.SR3	is the formatted output list of the time-sorted concentration parameters. Concentration contours are generated for the specified upper and lower levels of concern at the specified height entered in DEGADISIN or JETPLU_IN.
(*) RUN_NAME.TR4	contains the necessary information to recover the other output files to facilitate further processing.

```
! This is an example for an "ER3" run parameter file.
! The same rules apply as for the "ER1" files.
!
!234567890123456789012345678901234567890
!--------1---------2---------3
!
! These values are in common area /ERROR/
!
ERT1        20.       FIRST SORT TIME
ERDT        5.        SORT TIME DELTA
ERNTIM      20.       NUMBER OF TIMES FOR THE SORT
!
!    Note: ERNTIM is entered as a real variable even though
!          it is an integer type variable in the program.
!
!  The value of CHECK5 determines whether the above sort parameters
!     are used.  CHECK5 is initialized through the passed transfer
!     files to .FALSE.  CHECK5 is set to .TRUE. if a real value of 1.
!     is passed in this file.
!
CHECK5        0.       USE THE DEFAULT TIME PARAMETERS
!CHECK5       1.       USE THE TIME PARAMETERS GIVEN ABOVE
!
!
sigx_flag    1.       correction for x-direction dispersion is to be made
!sigx_flag    0.       no correction for x-direction dispersion
!
!
!
! End-of-File
```

Figure A.11. SYS$DEGADIS:EXAMPLE.ER3 Listing.

DEAGDIS4 sorts the downwind dispersion calculation for each of several observers for concentration information at several given positions; the along-wind dispersion correction is then applied as desired. DEGADIS4 uses the following subroutines (Figure A.12):

(*) DEGADIS4	contains the program overhead and sequentially calls the routines to recover the information generated in DEGADIS2, recover the numerical parameter file RUN_NAME.ER3, sort and apply the along-wind dispersion correction to the results of DEGADIS2, and output the results (D-33).
(*) DOSOUT	generates the formatted output file RUN_NAME.SR4 (D-45).
(*) ESTRT3	recovers the numerical parameters contained in the file RUN_NAME.ER3, particularly whether the x-direction dispersion correction is to be applied (D-58).
(*) GAMMA	calculates the gamma function of the argument (i.e. $\Gamma(x)$) (D-59).
(*) GETTIMDOS	sets the time sort parameters as required to output the concentration time history at the desired positions (D-63).
(*) INCGAMMA	calculates the incomplete gamma function of the two arguments (D-70).
(*) LIMIT	establishes the limit for ZBRENT (D-93).
(*) SERIES	evaluates a series needed to estimate the mass of gas above a given concentration level (D-130).
(*) SORTS	recovers the information in RUN_NAME.PSD and arranges the information according to the time sort parameters (D-134).
(*) SORTS1	applies the x-direction dispersion correction to the time-sorted information (D-131)
(*) STRT3	recovers the information generated in DEGADIS2 contained in the file RUN_NAME.TR3 (D-178).
(*) TPROP	is a series of routines which estimate the thermodynamic properties of a given gas mixture (D-185).
(*) TRANS3	writes RUN_NAME.TR4 which contains the necessary information to recover the other output files for this simulation (D-206).

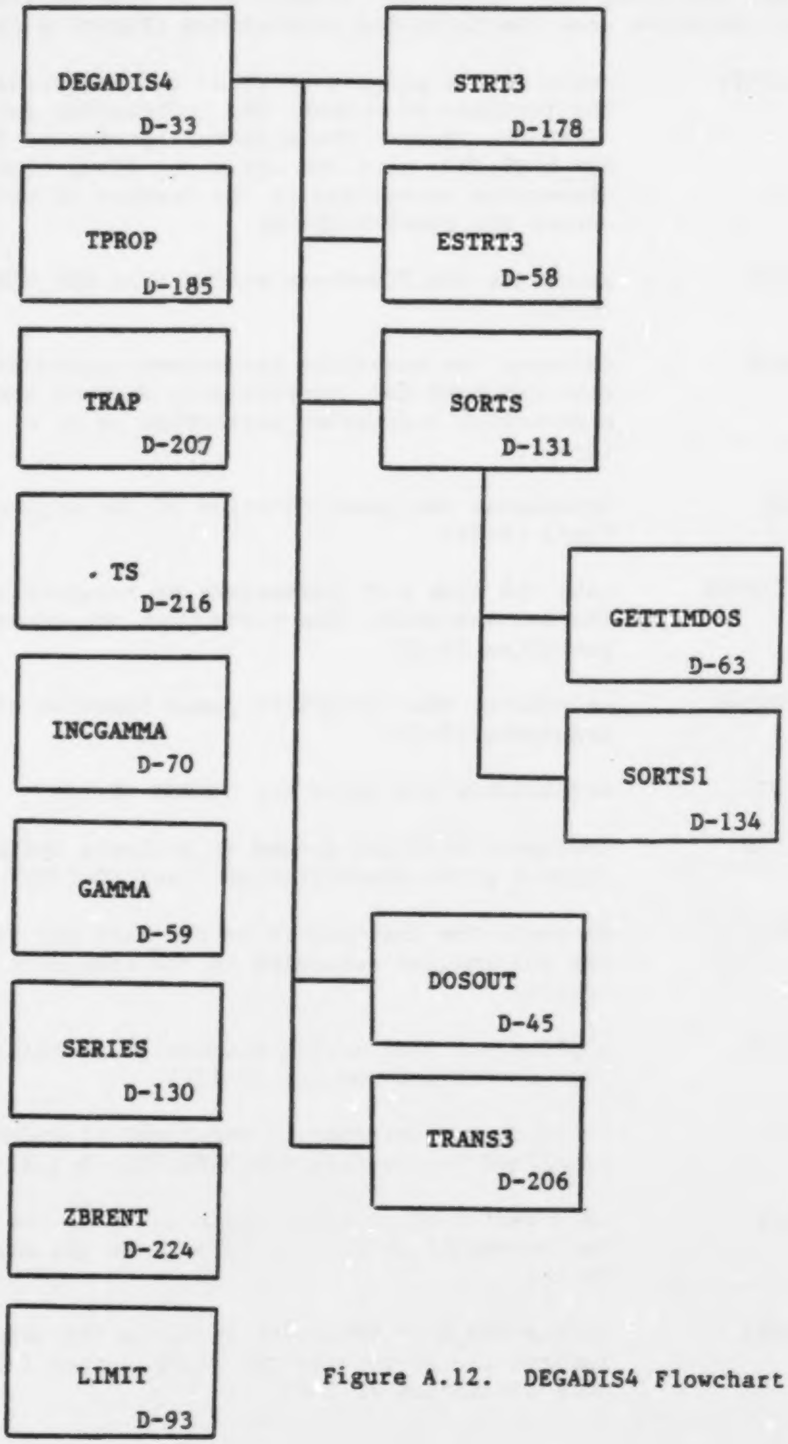

DEGADIS4
D-33
TPROP
D-185
TRAP
D-207
TS
D-216
INCGAMMA
D-70
GAMMA
D-59
SERIES
D-130
ZBRENT
D-224
LIMIT
D-93
STRT3
D-178
ESTRT3
D-58
SORTS
D-131
GETTIMDOS
D-63
SORTS1
D-134
DOSOUT
D-45
TRANS3
D-206

Figure A.12. DEGADIS4 Flowchart

(*) TRAP	is a utility included for program diagnostics (D-207).
(*) TS	calculates the time when a given observer will be at a given downwind distance (D-216).
(*) ZBRENT	finds the bracketed root of an equation using Brent's method (D-224).

As input, DEGADIS4 requires three files and input from the terminal:

(*) RUN_NAME.ER3	contains the flag which dictates whether the x-direction dispersion correction is applied. A copy of SYS$DEGADIS:EXAMPLE.ER3 file includes the x-direction dispersion correction which should apply for most simulations.
(*) RUN_NAME.PSD	contains the calculated downwind dispersion parameters for each observer. DEGADIS4 sorts this information to determine the downwind concentration time histories at the desired positions.
(*) RUN_NAME.TR3	contains the number of each record type written to RUN_NAME.PSD as well as the simulation definition.
(*) terminal input	DEGADIS4 prompts the user for the file name to be used for this run. In addition, DEGADIS4 requests the number of downwind distances (JDOS). For each downwind distance, DEGADIS4 asks for the x-position (DOSDISX(I)). Four positions are allowed for each downwind distance (DOSDISY(IJ,I)) and DOSDISZ(IJ,I) for IJ=1 to 4). If fewer than four positions are desired, negative or zero values are entered for the first position which is not desired. A summary of this input information is included in Figure A.13. Note that the same information can be put in a command file for batch processing. The same information can be put in a file which can be associated with FOR005.DAT so that a file can be used for input.

```
                        RUN_NAME
                        JDOS
                  ⎧     DOSDISX(I)
                  ⎪   ⎧ DOSDISY(1,I),DOSDISZ(1,I)
FOR I=1 to JDOS   ⎨   ⎪ DOSDISY(2,I),DOSDISZ(2,I)
                  ⎪  *⎨ DOSDISY(3,I),DOSDISZ(3,I)
                  ⎩   ⎩ DOSDISY(4,I),DOSDISZ(4,I)
```

Figure A.13. Structure of Input for DEGADIS4.

As output, DEGADIS4 generates two new files:

(*) RUN_NAME.SR4	is the formatted output list of the sorted concentration time histories.
(*) RUN_NAME.TR4	contains the necessary information to recover the other output files to facilitate further processing.

Steady-State Module--SDEGADIS2

SDEGADIS2 is a simplification of DEGADIS2 which uses many of the same subroutines. SDEGADIS2 performs the downwind dispersion portion of the calculation for a steady-state source described by DEGADIS1, SDEGADIS2 is composed of the following subroutines (Figure A.14):

(*) AFGEN	linearly interpolates between a pair of points based on a list of supplied values (D-3).
(*) GAMMA	calculates the gamma function of the argument (i.e. $\Gamma(x)$) (D-59).
(*) ESTRT2SS	recovers a subset of the numerical parameters contained in the file RUN_NAME.ER2 (D-56).
(*) INCGAMMA	calculates the incomplete gamma function of the two arguments (D-70).
(*) LIMIT	establishes the limit for ZBRENT (D-93).
(*) PSS	is the same subroutine used in DEGADIS2; it contains the ordinary differential equations which describe the downwind dispersion calculation when $b > 0$ (D-100).
(*) PSSOUTSS	governs the output of calculated points to the file RUN_NAME.SR3 when PSS is active (D-107).
(*) RIPHIF	calculates the Richardson number and the value of $\phi(Ri)$ (D-108).
(*) RKGST	performs numerical integration of a specified system of equations using a modified fourth-order Runge-Kutta methods (D-114).
(*) SDEGADIS2	contains the program overhead and sequentially calls the routines to recover the information generated in DEGADIS1, recover the numerical parameters file RUN_NAME.ER2, and perform the steady-state simulation (D-122).
(*) SERIES	evaluates a series needed to estimate the mass of gas above a given concentration level (D-130).

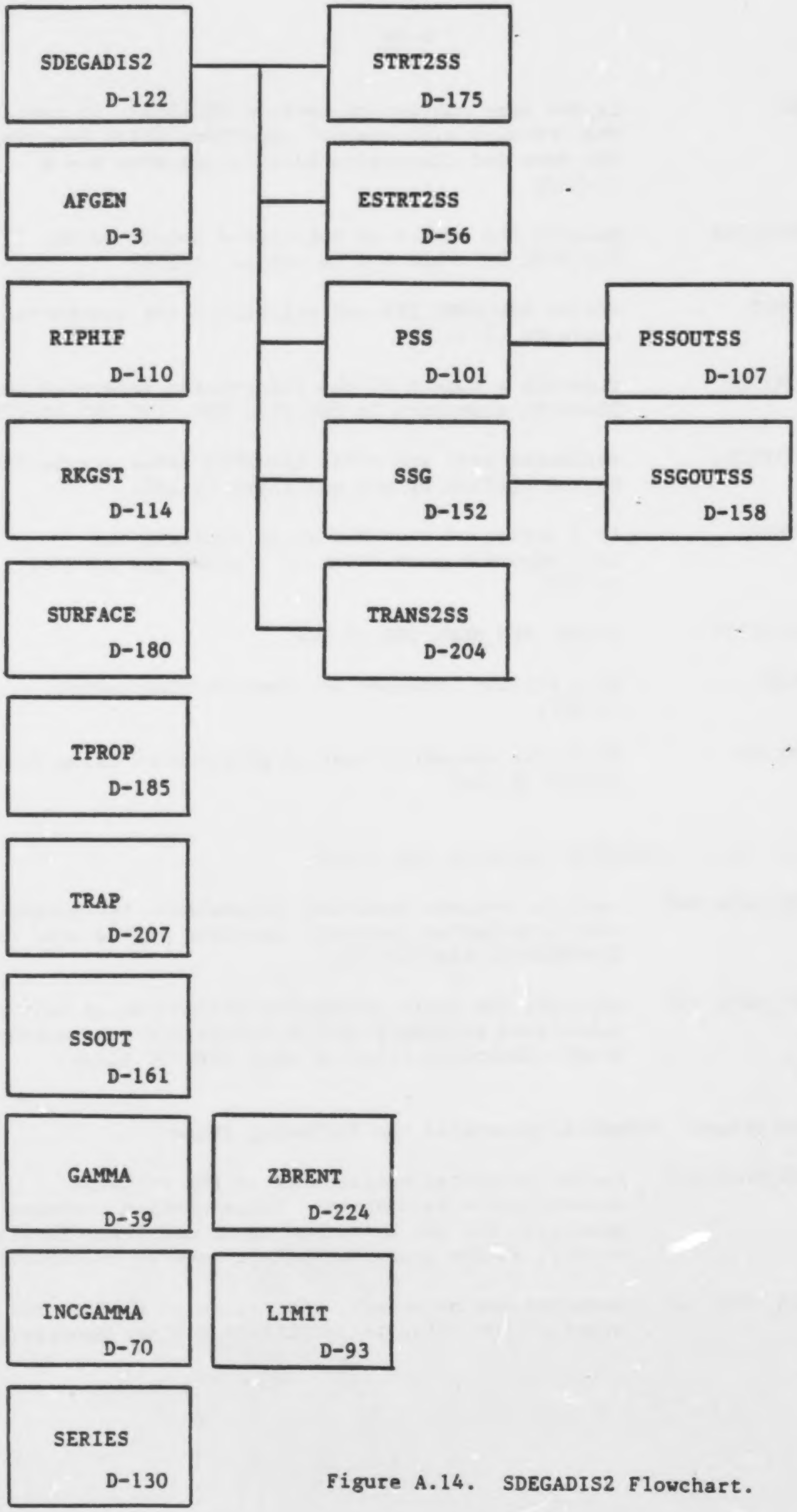

SDEGADIS2
D-122
STRT2SS
D-175
AFGEN
D-3
ESTRT2SS
D-56
RIPHIF
D-110
PSS
D-101
PSSOUTSS
D-107
RKGST
D-114
SSG
D-152
SSGOUTSS
D-158
SURFACE
D-180
TRANS2SS
D-204
TPROP
D-185
TRAP
D-207
SSOUT
D-161
GAMMA
D-59
ZBRENT
D-224
INCGAMMA
D-70
LIMIT
D-93
SERIES
D-130

Figure A.14. SDEGADIS2 Flowchart.

(*) SSG	is the same subroutine used in DEGADIS2; it contains the ordinary differential equations which describe the downwind dispersion calculation when b = 0 (D-152).
(*) SSGOUTSS	governs the output of calculated points to the file RUN_NAME.SR3 when SSG is active (D-158).
(*) SSOUT	writes RUN_NAME.SR3 and calculates the concentration contours (D-161).
(*) STRT2SS	recovers a subset of the information generated in DEAGDIS1 contained in the file RUN_NAME.TR2 (D-175).
(*) SURFACE	estimates heat and water transfer rates across the bottom surface of the gas layer (D-180).
(*) TPROP	is a series of routines which estimate the thermodynamic properties of a given gas mixture (D-185).
(*) TRANS2SS	writes RUN_NAME.TR3 (D-204).
(*) TRAP	is a utility included for program diagnostics (D-207).
(*) ZBRENT	finds the bracketed root of an equation using Brent's method (D-224).

As input, SDEAGDIS2 requires two files:

(*) RUN_NAME.ER2	contains various numerical parameters, the steady-state simulation typically requires only a copy of SYS$DEGADIS:EXAMPLE.ER2.
(*) RUN_NAME.TR2	contains the basic simulation definition as well as calculated secondary source parameters; the steady-state simulation requires only part of these.

As output, SDEAGDIS2 generates the following files:

(*) RUN_NAME.SR3	is the formatted output list of the downwind concentration parameters. Concentration contours are generated for the specified upper and lower levels of concern at the specified height entered in DEAGDISIN.
(*) RUN_NAME.TR3	contains the necessary information to recover the other output files to facilitate further processing.

APPENDIX B

EXAMPLE MODEL INPUT AND OUTPUT

Bhopal, India, MIC Release

The example conditions shown in Table B.1 for the accidental release of methylisocyanate (MIC) on December 3, 1984, in Bhopal, India, were reported by Singh (1986). MIC was assumed released as a pure, ambient temperature gas at a steady rate of 6.72 kg/s.

Figure B.1. MIC Example Simulation Release Conditions

Mass of MIC released	40 tons
Duration of release	90 min
Vent elevation	33 m
Vent diameter	0.2 m
Wind velocity at 10 m	1.9 m/s
Atmospheric stability	E/F

The input file for a steady-state release is shown in Figure B.1. A surface roughness of 0.1 m and an averaging time of 3600 s have been specified. For the lowest concentration of interest, the lethal concentrations to 50% of laboratory animals exposed to MIC (LC_{50}) for one and two hours are about 30 and 20 ppm, respectively. The concentrations of 30 ppm (GASUL=0.00003 mole fraction) and 20 ppm (GASLL=0.00002 mole fraction) are used to determine concentration contours in DEGADIS. The steady-state condition is indicated by TEND=0.

A batch command file is shown in Figure B.2 for the simulation called EX1. (The logical symbol SYS$DEGADIS: is assigned to the directory which contains the executable image of the model.) The output of the model is in the file with the original name and extension LIS for (for this example, EX1.LIS). To run the model interactively, the lines shown in Figure B.2 can be entered at a terminal.

The program JETPLU_IN reads the input file and generates the input file to the JETPLU program; when JETPLU_IN finishes, JETPLU_IN writes the line "JETPLU_IN - beginning command file". At this point, the JETPLU program begins; several lines of output are generated showing the values of various parameters calculated by the program. When JETPLU finishes, DEGBRIDGE begins; DEGBRIDGE takes the output of JETPLU and

```
Methylisocyanate (MIC) release simulation


2.9  10.                                  UO, ZO
0.1                                       ZR
1  6   0.                                 INDVEL, ISTAB, RML
298.  1.  50.                             TAMB, PAMB, RELHUM
298.                                      TSURF
MIC                                       GASNAM
57.                                       GASMW
3600.                                     AVTIME - based on 1 hr toxic levels
298.                                      JETTEM
0.00003   0.00002  0.5                    GASUL, GASLL, ZLL
0  2000.  0.                              INDHT,  CPK, CPP
-1                                        NDEN
6.72                                      ERATE
33. 0.2                                   ELEJET, DIAJET
0.0                                       TEND
100.                                      DISTMX

This is the end of the file.  Any comments can be included here since the
file is not read after the line for DISTMX above.
```

Figure B.1. Listing of EX1.IN.

```
$ RUN SYS$DEGADIS:JETPLU_IN
EX1
```

Figure B.2. Example Command File Used to Simulate the EX1 Simulation.

generates the necessary input file for DEGADIS. When DEGBRIDGE finishes, the standard DEGADIS procedure begins by copying the ER1 and ER2 parameter files. DEGADIS generates several lines of output showing the values of various numerical parameters calculated by the model. Last, the model generates the LIS file from the output files of JETPLU and DEGADIS (Figure B.3).

In the JETPLU model output, columns 3-5 correspond to plume centerline position. For the specified height, columns 9 through 11 correspond to the maximum concentration and horizontal distances to the specified mole fractions, respectively. For the MIC example case, the JETPLU calculations are continued until the centerline mole fraction drops below 5 ppm (GASLL/4). The model predicts the maximum extent of the 30 and 20 ppm concentration levels to be 2500 and 4200 m, respectively.

```
********************                JETPLU/DEGADIS v2.1                ********************

 6-SEP-1989 14:40:54.50

Methylisocyanate (MIC) release simulation



Ambient Meteorological Conditions...

Ambient windspeed at reference height:    2.9000     m/s
                     Reference height:    10.000     m
                    Surface roughness:    0.10000    m

               Pasquill stability class:              F

                  Monin-Obukhov length:   17.524     m
                     Friction velocity:   0.13910    m/s
                   Ambient temperature:   298.00     K
                      Ambient pressure:   1.0000     atm
                      Ambient humidity:  1.00882E-02
                     Relative humidity:   50.000     %

              Specified averaging time:   3600.0     s
                                DELTAy:  9.64473E-02
                                 BETAy:  0.90000
                                DELTAz:  1.12200E-02
                                 BETAz:   1.4024
                                GAMMAz: -5.40000E-02

Contaminant Properties...

        Contaminant molecular weight:    57.000
                 Initial temperature:    298.00
                Upper level of interest: 3.00000E-05
                Lower level of interest: 2.00000E-05
               Heat capacity constant:   80700.
                  Heat capacity power:   1.0000

NDEN flag:  -1

ISOFL flag:   1

Release Properties...

                        Release rate:   6.7200     kg/s
                  Discharge elevation:  33.000     m
                   Discharge diameter:  0.20000    m

Model Parameters...

                               ALFA1:  2.80000E-02
                               ALFA2:  0.37000
                              DISTMX:  100.00     m
```

Figure B.3. LIS File from the Output Files of JETPLU and DEGADIS.

```
                                                                                                  ------- At z=   0.500     m -----
Downwind  Elevation       Mole    Centerline  Density  Temperature  Sigma y     Sigma z        Mole            Width to:
Distance                Fraction Concentration                                               Fraction 2.00E-03molI 3.00E-03m
  (m)        (m)                  (kg/m3)     (kg/m3)       (K)        (m)         (m)                     (m)          (m)
.
4.122E-03    34.4        1.00        2.33       2.33        298.       0.110       0.110      0.000E+00
  56.9       45.8     8.888E-03   2.072E-02     1.19        298.        4.03        2.14      0.000E+00
  110.       45.8     3.728E-03   8.690E-03     1.18        298.        6.87        3.02      0.000E+00

  178.       45.0     1.736E-03   4.047E-03     1.18        298.        10.4        4.28      0.000E+00
  257.       44.1     9.506E-04   2.216E-03     1.18        298.        14.4        5.66      0.000E+00
  327.       43.2     6.391E-04   1.490E-03     1.18        298.        17.9        6.79      0.000E+00

  417.       42.3     4.304E-04   1.003E-03     1.18        298.        22.2        8.12      0.000E+00
  517.       41.3     3.039E-04   7.086E-04     1.18        298.        27.0        9.49      0.000E+00
  617.       40.3     2.289E-04   5.337E-04     1.18        298.        31.6        10.8      0.000E+00

  717.       39.4     1.803E-04   4.205E-04     1.18        298.        36.1        11.9      0.000E+00
  817.       38.6     1.468E-04   3.423E-04     1.18        298.        40.6        13.0      0.000E+00
  917.       37.7     1.225E-04   2.857E-04     1.18        298.        45.0        14.1      0.000E+00

1.017E+03    36.9     1.043E-04   2.432E-04     1.18        298.        49.4        15.1      0.000E+00
1.117E+03    36.2     9.023E-05   2.104E-04     1.18        298.        53.8        16.0      0.000E+00
1.21"E+03    35.4     7.909E-05   1.844E-04     1.18        298.        58.1        16.9      1.788E-05

1.317E+03    34.6     7.010E-05   1.634E-04     1.18        298.        62.3        17.8      2.119E-05   21.2
1.417E+03    33.9     6.275E-05   1.463E-04     1.18        298.        66.5        18.7      2.405E-05   40.4
1.517E+03    33.2     5.667E-05   1.321E-04     1.18        298.        70.7        19.5      2.642E-05   52.8

1.617E+03    32.5     5.161E-05   1.203E-04     1.18        298.        74.9        20.2      2.831E-05   62.4
1.717E+03    31.8     4.736E-05   1.104E-04     1.18        298.        79.1        21.0      2.976E-05   70.5
1.817E+03    31.1     4.378E-05   1.021E-04     1.18        298.        83.2        21.7      3.083E-05   77.4         19.4

1.917E+03    30.5     4.075E-05   9.500E-05     1.18        298.        87.3        22.4      3.156E-05   83.4         27.8
2.017E+03    29.8     3.818E-05   8.901E-05     1.18        298.        91.3        23.1      3.201E-05   88.6         32.9
2.117E+03    29.2     3.599E-05   8.391E-05     1.18        298.        95.4        23.8      3.223E-05   93.2         36.1

2.217E+03    28.6     3.411E-05   7.953E-05     1.18        298.        99.4        24.4      3.225E-05   97.2         37.8
2.317E+03    28.0     3.250E-05   7.578E-05     1.18        298.        103.        25.0      3.212E-05   101.         38.2
2.417E+03    27.4     3.111E-05   7.252E-05     1.18        298.        107.        25.6      3.186E-05   104.         37.3

2.517E+03    26.9     2.989E-05   6.968E-05     1.18        298.        111.        26.2      3.151E-05   106.         34.9
2.617E+03    26.3     2.882E-05   6.719E-05     1.18        298.        115.        26.8      3.108E-05   108.         30.7
2.717E+03    25.8     2.788E-05   6.499E-05     1.18        298.        119.        27.4      3.060E-05   110.         23.7

2.817E+03    25.2     2.703E-05   6.302E-05     1.18        298.        123.        27.9      3.007E-05   111.         8.22
2.917E+03    24.7     2.627E-05   6.124E-05     1.18        298.        127.        28.4      2.951E-05   112.
3.017E+03    24.2     2.557E-05   5.962E-05     1.18        298.        131.        29.0      2.892E-05   113.

3.117E+03    23.7     2.494E-05   5.814E-05     1.18        298.        135.        29.5      2.833E-05   113.
3.217E+03    23.2     2.435E-05   5.676E-05     1.18        298.        139.        30.0      2.773E-05   112.
3.317E+03    22.7     2.380E-05   5.548E-05     1.18        298.        143.        30.5      2.712E-05   111.

3.417E+03    22.2     2.328E-05   5.427E-05     1.18        298.        147.        31.0      2.652E-05   110.
3.517E+03    21.7     2.279E-05   5.313E-05     1.18        298.        150.        31.4      2.592E-05   108.
3.617E+03    21.2     2.232E-05   5.204E-05     1.18        298.        154.        31.9      2.533E-05   106.

3.717E+03    20.8     2.188E-05   5.100E-05     1.18        298.        158.        32.3      2.475E-05   103.
3.817E+03    20.3     2.145E-05   5.000E-05     1.18        298.        162.        32.8      2.418E-05   99.8
3.917E+03    19.9     2.103E-05   4.904E-05     1.18        298.        166.        33.2      2.362E-05   95.6

4.017E+03    19.4     2.063E-05   4.810E-05     1.18        298.        169.        33.7      2.308E-05   90.7
4.117E+03    19.0     2.025E-05   4.720E-05     1.18        298.        173.        34.1      2.255E-05   84.9
4.217E+03    18.6     1.987E-05   4.632E-05     1.18        298.        177.        34.5      2.204E-05   78.0
```

Figure B.3. (continued)

```
4.317E+03   18.2        1.950E-05  4.547E-05   1.18        298.        181.        34.9        2.153E-05   69.5
4.417E+03   17.7        1.915E-05  4.463E-05   1.18        298.        185.        35.3        2.105E-05   58.9
4.517E+03   17.3        1.880E-05  4.382E-05   1.18        298.        188.        35.7        2.057E-05   44.7

4.617E+03   16.9        1.846E-05  4.303E-05   1.18        298.        192.        36.1        2.011E-05   20.4
4.717E+03   16.5        1.812E-05  4.226E-05   1.18        298.        196.        36.5        1.967E-05
4.817E+03   16.1        1.780E-05  4.150E-05   1.18        298.        200.        36.8        1.923E-05

4.917E+03   15.7        1.748E-05  4.076E-05   1.18        298.        203.        37.2        1.882E-05
5.017E+03   15.4        1.717E-05  4.004E-05   1.18        298.        207.        37.6        1.841E-05
5.117E+03   15.0        1.687E-05  3.933E-05   1.18        298.        211.        37.9        1.802E-05

5.217E+03   14.6        1.657E-05  3.864E-05   1.18        298.        214.        38.3        1.763E-05
5.317E+03   14.2        1.628E-05  3.796E-05   1.18        298.        218.        38.6        1.726E-05
5.417E+03   13.8        1.600E-05  3.730E-05   1.18        298.        222.        39.0        1.690E-05

5.517E+03   13.5        1.572E-05  3.665E-05   1.18        298.        225.        39.3        1.656E-05
5.617E+03   13.1        1.545E-05  3.602E-05   1.18        298.        229.        39.6        1.622E-05
5.717E+03   12.8        1.518E-05  3.540E-05   1.18        298.        233.        40.0        1.589E-05

5.817E+03   12.4        1.492E-05  3.479E-05   1.18        298.        236.        40.3        1.558E-05
5.917E+03   12.1        1.467E-05  3.420E-05   1.18        298.        240.        40.6        1.527E-05
6.017E+03   11.7        1.442E-05  3.362E-05   1.18        298.        244.        40.9        1.497E-05

6.117E+03   11.4        1.418E-05  3.306E-05   1.18        298.        247.        41.3        1.468E-05
6.217E+03   11.0        1.394E-05  3.250E-05   1.18        298.        251.        41.6        1.440E-05
6.317E+03   10.7        1.371E-05  3.196E-05   1.18        298.        254.        41.9        1.413E-05

6.417E+03   10.4        1.348E-05  3.143E-05   1.18        298.        258.        42.2        1.387E-05
6.517E+03   10.0        1.326E-05  3.091E-05   1.18        298.        262.        42.5        1.361E-05
6.617E+03   9.71        1.304E-05  3.041E-05   1.18        298.        265.        42.8        1.336E-05

6.717E+03   9.38        1.283E-05  2.991E-05   1.18        298.        269.        43.1        1.312E-05
6.817E+03   9.06        1.262E-05  2.943E-05   1.18        298.        273.        43.3        1.289E-05
6.917E+03   8.74        1.242E-05  2.895E-05   1.18        298.        276.        43.6        1.266E-05

7.017E+03   8.43        1.222E-05  2.849E-05   1.18        298.        280.        43.9        1.244E-05
7.117E+03   8.12        1.203E-05  2.804E-05   1.18        298.        283.        44.2        1.222E-05
7.217E+03   7.81        1.184E-05  2.759E-05   1.18        298.        287.        44.5        1.201E-05

7.317E+03   7.50        1.165E-05  2.716E-05   1.18        298.        290.        44.7        1.181E-05
7.417E+03   7.19        1.147E-05  2.674E-05   1.18        298.        294.        45.0        1.161E-05
7.517E+03   6.89        1.129E-05  2.632E-05   1.18        298.        298.        45.3        1.142E-05

7.617E+03   6.59        1.112E-05  2.592E-05   1.18        298.        301.        45.5        1.123E-05
7.717E+03   6.30        1.095E-05  2.552E-05   1.18        298.        305.        45.8        1.105E-05
7.817E+03   6.00        1.078E-05  2.514E-05   1.18        298.        308.        46.0        1.087E-05

7.917E+03   5.71        1.062E-05  2.476E-05   1.18        298.        312.        46.3        1.070E-05
8.017E+03   5.42        1.046E-05  2.439E-05   1.18        298.        315.        46.5        1.053E-05
8.117E+03   5.13        1.031E-05  2.403E-05   1.18        298.        319.        46.8        1.037E-05

8.217E+03   4.84        1.015E-05  2.367E-05   1.18        298.        322.        47.0        1.021E-05
8.317E+03   4.56        1.000E-05  2.332E-05   1.18        298.        326.        47.3        1.005E-05
8.417E+03   4.28        9.859E-06  2.298E-05   1.18        298.        329.        47.5        9.898E-06

8.517E+03   4.00        9.716E-06  2.265E-05   1.18        298.        333.        47.8        9.750E-06
8.617E+03   3.72        9.577E-06  2.233E-05   1.18        298.        336.        48.0        9.605E-06
8.717E+03   3.45        9.440E-06  2.201E-05   1.18        298.        340.        48.2        9.464E-06

8.817E+03   3.17        9.307E-06  2.170E-05   1.18        298.        343.        48.5        9.326E-06
8.917E+03   2.90        9.176E-06  2.139E-05   1.18        298.        347.        48.7        9.192E-06
9.017E+03   2.63        9.048E-06  2.109E-05   1.18        298.        350.        48.9        9.061E-06

9.117E+03   2.37        8.923E-06  2.080E-05   1.18        298.        354.        49.2        8.933E-06
9.217E+03   2.10        8.800E-06  2.052E-05   1.18        298.        357.        49.4        8.808E-06
9.317E+03   1.84        8.680E-06  2.024E-05   1.18        298.        361.        49.6        8.685E-06

9.417E+03   1.58        8.562E-06  1.996E-05   1.18        298.        364.        49.8        8.566E-06
9.517E+03   1.32        8.447E-06  1.969E-05   1.18        298.        368.        50.0        8.449E-06
9.617E+03   1.06        8.334E-06  1.943E-05   1.18        298.        371.        50.3        8.335E-06

9.717E+03   0.803       8.223E-06  1.917E-05   1.18        298.        375.        50.5        8.224E-06
9.817E+03   0.549       8.115E-06  1.892E-05   1.18        298.        378.        50.7        8.115E-06
9.917E+03   0.296       8.009E-06  1.867E-05   1.18        298.        382.        50.9        8.008E-06

1.002E+04   4.518E-02   7.905E-06  1.843E-05   1.18        298.        385.        51.1        7.904E-06
1.003E+04   0.000E+00   7.886E-06  1.839E-05   1.18        298.        386.        51.1        7.886E-06
```

Figure B.3. (continued)

```
*********************       U O A _ D E G A D I S   M O D E L   O U T P U T   - -   V E R S I O N   2.1       *****************
****************
                                        *************** 6-SEP-1989 14:44:19.11 ***************
Data input on                                                                                                    ************
Source program run on              6-SEP-1989 14:44:11.16
                                   6-SEP-1989 14:44:19.11

          TITLE BLOCK

Methylisocyanate (MIC) release simulation



     Wind velocity at reference height
     Reference height                                        2.90   m/s
                                                            10.00   m
     Surface roughness length
                                                         0.100      m
     Pasquill Stability class
                                                           F
     Monin-Obukhov length
     Gaussian distribution constants                     17.5       m
                     Specified averaging time
                                        Delta             3600.00   s
                                         Beta             0.09645
                                                          0.90000
     Wind velocity power law constant    Alpha
     Friction velocity                                    0.44904
                                                          0.13910   m/s
     Ambient Temperature
     Ambient Pressure                                     298.00    K
     Ambient Absolute Humidity                             1.000    atm
     Ambient Relative Humidity                          1.008E-02   kg/kg BDA
                                                           50.00    %
     Input:          Mole fraction      CONCENTRATION OF C      GAS DENSITY
                                             kg/m**3              kg/m**3
                        0.00000              0.00000              1.17737
                        1.00000              2.33112              2.33112

     Specified Gas Properties:

         Molecular weight:
         Storage temperature:                                       57.000
         Density at storage temperature and ambient pressure:       298.00     K
         Mean heat capacity constant:                               2.3311     kg/m**3
         Mean heat capacity power:                                  80700.
         Upper mole fraction contour:                               1.0000
         Lower mole fraction contour:                              3.00000E-05
         Height for isopleths:                                     2.00000E-05
                                                                    0.50000    m
Source input data points

           Initial mass in cloud:    0.00000E+00

     Time         Contaminant      Source Radius     Contaminant       Temperature        Enthalpy
                   Mass Rate                         Mass Fraction
      s             kg/s               m           kg contam/kg mix         K                J/kg
 0.00000E+00       6.7200            829.28          1.56157E-05          298.00         0.00000E+00
  60230.           6.7200            829.28          1.56157E-05          298.00         0.00000E+00
  60231.         0.00000E+00       0.00000E+00       1.56157E-05          298.00         0.00000E+00
  60232.         0.00000E+00       0.00000E+00       1.56157E-05          298.00         0.00000E+00

     Calculation procedure for ALPHA:  1

     Entrainment prescription for PHI:  3

     Layer thickness ratio used for average depth:     2.1500

     Air entrainment coefficient used: 0.590

     Gravity slumping velocity coefficient used: 1.150

     Isothermal calculation

     Heat transfer not included
```

Figure B.3. (continued)

```
           *****                           CALCULATED SOURCE PARAMETERS                          *****
   Time        Gas Radius       Height          Qstar        SZ(x=L/2.)    Mole frac C        Density       Rich No.
   sec             m               m          kg/m**2/s          m                           kg/m**3

 0.000000E+00    829.280      1.100000E-05   2.741708E-06    56.6987      7.886120E-06       1.17738       0.756144
  5.75485        829.293      0.115404       2.737199E-06    61.9269      7.873082E-06       1.17738       0.756144
Source strength [kg/s] :                         6.7200
Equivalent Primary source length [m] :           1658.6     Equivalent Primary source radius [m] :              829.28
                                                            Equivalent Primary source half-width [m] :          651.32
Secondary source concentration [kg/m**3] :    1.83552E-05  Secondary source SZ [m] :                           61.927
Contaminant flux rate:    3.11030E-06


Secondary source mass fractions... contaminant:  1.558989E-05   air:   0.99000
          Enthalpy:    0.00000E+00       Density:     1.1774

Secondary source length [m] :                    1658.6     Secondary source half-width [m] :                   651.33

Distance     Mole    Concentration Density  Temperature   Half          Sz          Sy      Width at z=  0.50 m to:
           Fraction                                        Width                            2.000E-03mole% 3.000E-03mole%
  (m)                  (kg/m**3)  (kg/m**3)     (K)         (m)         (m)         (m)          (m)           (m)

1.086E+04  7.873E-06   1.836E-05    1.18       298.        651.        61.9      0.000E+00
1.086E+04  7.873E-06   1.836E-05    1.18       298.        650.        61.9        1.86
1.086E+04  7.873E-06   1.835E-05    1.18       298.        649.        61.9        2.64

1.087E+04  7.872E-06   1.835E-05    1.18       298.        635.        62.0        18.8
1.087E+04  7.861E-06   1.833E-05    1.18       298.        628.        62.0        26.5

For the UFL of   3.00000E-03 mole percent, and the LFL of   2.00000E-03 mole percent:

The mass of contaminant between the UFL and LFL is:  0.00000E+00 kg.
The mass of contaminant above the LFL is:   0.00000E+00 kg.
```

Figure B.3. (concluded)

Enid, Oklahoma, Ammonia Release

The model was also used to simulate an ammonia pipeline failure which occurred near Enid, Oklahoma, on May 7, 1976 (Atwood, 1976). The release occurred during early morning hours (~ 8 AM) when a bulldozer struck and ruptured an eight-inch ammonia pipeline operating at approximately 700 psi. Approximately 500 tons of ammonia were released in 4.5 hours from the ten-mile section isolated between check valves. The windspeed at the time of the accident was reported to be approximately 10 mph. The dense aerosol plume reportedly etched a parabolic-shaped scar about 6 miles long and one-half mile wide. The conditions used to simulate the release are shown in Table B.2.

Table B.2. Ammonia Example Simulation Release Conditions

Ammonia release rate	56 kg/s (twice 500 tons per 4.5 hours
Release elevation	zero (ground level)
Vent diameter	0.2 m
Wind speed (at 10 m)	4.5 m/s
Surface roughness	1 cm
Atmospheric stability	D

The input file is shown in Figure B.4. The release elevation has been set to 0.02 m (2 z_R) to avoid numerical problems. The ammonia aerosol/(humid) air mixture density shown in Figure B.5 was determined by assuming the ambient air and release ammonia aerosol mixes adiabatically (Spicer and Havens, 1988). The relationship between mixture density and ammonia concentration is input to the model with 20 ordered triples of ammonia mole fraction, ammonia concentration (kg/m^3), and ammonia/air mixture density (kg/m^3).

Figure B.6 shows the predicted 1000 ppm and 100 ppm ground-level concentration isopleths which assume one-hour averaging time for the estimation of lateral dispersion by meander. The predicted plume rise was approximately 70 meters, and the plume was predicted to return to ground level about 525 meters downwind of the release. The LIS file from the output files of JETPLU and DEGADIS is shown in Figure B.7.

```
Enid, Oklahoma, ammonia pipeline release simulation

 4.5  10.
 0.01                                              UO, ZO
 1  4    0.                                        ZR
 298.  0.98  50.                                   INDVEL, ISTAB, RML
 298.                                              TAMB, PAMB, RELHUM
 NH3                                               TSURF
 17.                                               GASNAM
 3600.                                             GASMW
 298.                                              AVTIME
 0.001  0.0001  0.                                 TEMJET
 0  3845.000  1.000000                             GASUL, GASLL, ZLL
  22                                               INDHT,  CPK, CPP
   0.0000000E+00      0.0000000E+00                NDEN
   1.4000000E-02      9.8185645E-03                1.153908
   2.0000000E-02      1.4134786E-02                1.185840
   4.2000000E-02      3.0499677E-02                1.192022
   6.2000000E-02      4.6361663E-02                1.213623
   8.1000000E-02      6.2597632E-02                1.239219
   0.1050000          8.5226245E-02                1.270430
   0.1420000          0.1255372                    1.320677
   0.1610000          0.1486259                    1.415537
   0.1760000          0.1675721                    1.465818
   0.1890000          0.1841250                    1.501815
   0.2040000          0.2030910                    1.527787
   0.2230000          0.2269812                    1.550787
   0.4430000          0.5594106                    1.571988
   0.5560000          0.8019109                    1.755602
   0.6540000           1.081982                    1.890973
   0.7390000           1.410169                    2.055484
   0.8120000           1.797091                    2.257174
   0.8740000           2.255920                    2.504694
   0.9250000           2.790716                    2.809017
   0.9670000           3.424175                    3.175532
    1.000000           4.149494                    3.622905
 56.                                               4.149494
 0.02 0.2                                          ERATE
 0.0                                               ELEJET, DIAJET
 20.                                               TEND
                                                   DISTMX

This is the end of the file.  Any comments can be included here since
the file is not read after the line for DISTMX above.
```

Figure B.4. Listing of EX2.IN.

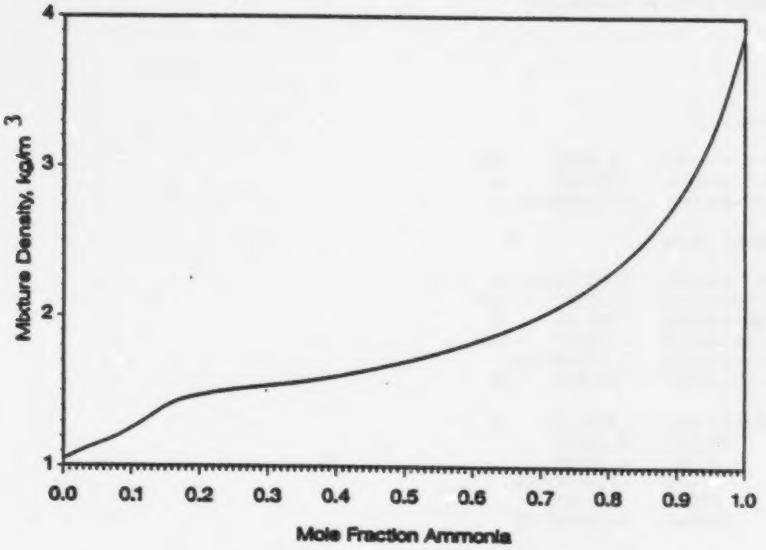


Figure B.5. Ammonia Aerosol/Air Adiabatic Mixture Density.

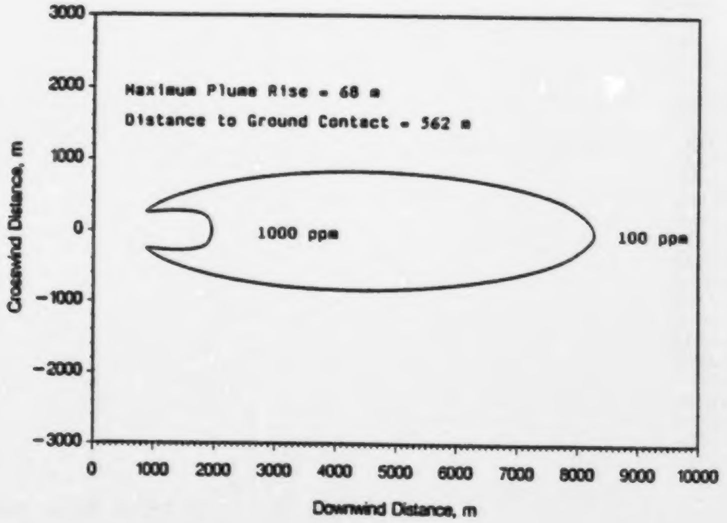


Figure B.6. Ground-Level Isocontours--Oklahoma Ammonia Pipeline Break

```
********************                    JETPLU/DEGADIS v2.1                    ********************
 6-SEP-1989 15:48:31.06
Enid, Oklahoma, ammonia pipeline release simulation



Ambient Meteorological Conditions...

Ambient windspeed at reference height:    4.5000      m/s
                     Reference height:    10.000      m
                    Surface roughness:   1.00000E-02  m

              Pasquill stability class:              D

                  Monin-Obukhov length:  0.00000E+00  m
                     Friction velocity:  0.22797      m/s
                   Ambient temperature:   298.00      K
                      Ambient pressure:  0.98000      atm
                      Ambient humidity:  1.03009E-02
                     Relative humidity:   50.000      %

              Specified averaging time:   3600.0      s
                                DELTAy:  0.19461
                                 BETAy:  0.90000
                                DELTAz:  4.13400E-02
                                 BETAz:   1.1737
                                GAMMAz: -3.16000E-02

Contaminant Properties...

        Contaminant molecular weight:   17.000
                 Initial temperature:   298.00
              Upper level of interest:  1.00000E-03
              Lower level of interest:  1.00000E-04
              Heat capacity constant:   3845.0
                 Heat capacity power:   1.0000

NDEN flag:  22
 Mole fraction  Concentration    Density
                   (kg/m3)       (kg/m3)
   0.00000E+00   0.00000E+00    1.1539
   1.40000E-02   9.81856E-03    1.1858
   2.00000E-02   1.41348E-02    1.1920
   4.20000E-02   3.04997E-02    1.2136
   6.20000E-02   4.63617E-02    1.2392
   8.10000E-02   6.25976E-02    1.2704
   0.10500       8.52262E-02    1.3207
   0.14200       0.12554        1.4155
   0.16100       0.14863        1.4658
   0.17600       0.16757        1.5018
   0.18900       0.18413        1.5278
   0.20400       0.20309        1.5508
   0.22300       0.22698        1.5720
   0.44300       0.55941        1.7556
   0.55600       0.80191        1.8910
   0.65400        1.0820        2.0555
   0.73900        1.4102        2.2572
   0.81200        1.7971        2.5047
   0.87400        2.2559        2.8090
   0.92500        2.7907        3.1755
   0.96700        3.4242        3.6229
    1.0000        4.1495        4.1495

ISOFL flag:   1


Release Properties...

                   Release rate:   56.000      kg/s
            Discharge elevation:   2.00000E-02 m
             Discharge diameter:   0.20000     m


Model Parameters...

                          ALFA1:   2.80000E-02
```

Figure B.7. LIS File from Output Files of JETPLU and DEGADIS.

```
Downwind  Elevation     Mole      Centerline    Density  Temperature  Sigma y    Sigma z   ------- At z=   0.000E+00 m ----
Distance              Fraction  Concentration                                                Mole           Width to:
  (m)        (m)                  (kg/m3)      (kg/m3)       (K)        (m)        (m)      Fraction 1.00E-02molZ 1.00E-01r
                                                                                                        (m)          (m)
2.151E-17    1.56       1.00        4.15         4.15        298.       0.110      0.110   0.000E+00
  13.2       45.9    6.181E-02   4.642E-02       1.24        298.       5.00       4.64    0.000E+00
  27.5       56.0    4.044E-02   2.944E-02       1.21        298.       7.36       6.44    0.000E+00

  41.3       61.2    3.127E-02   2.251E-02       1.20        298.       9.14       7.56    0.000E+00
  60.9       65.3    2.423E-02   1.729E-02       1.20        298.       11.2       8.57    0.000E+00
  80.8       67.2    2.004E-02   1.423E-02       1.19        298.       13.1       9.18    0.000E+00

  101.       67.5    1.724E-02   1.220E-02       1.19        298.       14.9       9.54    0.000E+00
  121.       66.6    1.460E-02   1.029E-02       1.19        298.       16.8       10.1    0.000E+00
  141.       64.7    1.219E-02   8.553E-03       1.18        298.       18.8       10.8    0.000E+00

  160.       62.2    1.019E-02   7.112E-03       1.18        298.       20.9       11.7    0.000E+00
  180.       59.4    8.577E-03   5.965E-03       1.17        298.       23.1       12.6    0.000E+00
  200.       56.3    7.297E-03   5.060E-03       1.17        298.       25.3       13.6    0.000E+00

  220.       53.2    6.278E-03   4.343E-03       1.17        298.       27.4       14.6    0.000E+00
  240.       50.0    5.459E-03   3.769E-03       1.17        298.       29.6       15.7    0.000E+00
  259.       46.7    4.794E-03   3.304E-03       1.16        298.       31.7       16.7    0.000E+00

  279.       43.5    4.246E-03   2.923E-03       1.16        298.       33.8       17.7    0.000E+00
  299.       40.2    3.791E-03   2.607E-03       1.16        298.       35.9       18.7    7.628E-04    72.5
  318.       37.0    3.411E-03   2.343E-03       1.16        298.       38.0       19.8    1.183E-03    81.8         22.1

  338.       33.8    3.098E-03   2.127E-03       1.16        298.       40.1       20.7    1.638E-03    86.2         39.9
  358.       30.7    2.857E-03   1.960E-03       1.16        298.       42.2       21.7    2.077E-03    90.7         51.1
  378.       27.5    2.700E-03   1.852E-03       1.16        298.       44.2       22.7    2.461E-03    95.1         59.5

  397.       24.4    2.635E-03   1.807E-03       1.16        298.       46.3       23.7    2.764E-03    99.5         66.1
  417.       21.4    2.653E-03   1.819E-03       1.16        298.       48.3       24.6    2.978E-03    104.         71.5
  437.       18.3    2.728E-03   1.871E-03       1.16        298.       50.4       25.5    3.105E-03    108.         76.0

  457.       15.3    2.821E-03   1.935E-03       1.16        298.       52.4       26.5    3.155E-03    113.         79.6
  477.       12.4    2.896E-03   1.987E-03       1.16        298.       54.4       27.4    3.141E-03    117.         82.4
  496.       9.48    2.928E-03   2.009E-03       1.16        298.       56.4       28.3    3.077E-03    121.         84.7

  516.       6.60    2.904E-03   1.993E-03       1.16        298.       58.3       29.1    2.975E-03    125.         86.3
  536.       3.75    2.826E-03   1.939E-03       1.16        298.       60.3       30.0    2.848E-03    130.         87.4
  556.       0.937   2.703E-03   1.854E-03       1.16        298.       62.3       30.9    2.704E-03    134.         88.0

  562.    0.000E+00  2.654E-03   1.820E-03       1.16        298.       62.9       31.2    2.653E-03    135.         88.1

                                              (END OF JETPLU SIMULATION)
```

Figure B.7. (continued)

```
*********************        U O A _ D E G A D I S   M O D E L   O U T P U T   - -   V E R S I O N   2.1        *****************
****************
                                        ***************  6-SEP-1989 15:49:46.45  ***************
Data input on                                                                                                     ***********
Source program run on               6-SEP-1989 15:49:38.76
                                    6-SEP-1989 15:49:46.45

      TITLE BLOCK

Enid, Oklahoma ammonia pipeline release simulation



    Wind velocity at reference height
    Reference height                                      4.50  m/s
                                                         10.00  m
    Surface roughness length
                                                     1.000E-02  m
    Pasquill Stability class
                                                      D
    Monin-Obukhov length
    Gaussian distribution constants                   infinite
                      Specified averaging time
                                        Delta          3600.00  s
                                         Beta          0.19461
                                                       0.90000
    Wind velocity power law constant    Alpha
    Friction velocity                                  0.18680
                                                       0.22797  m/s
    Ambient Temperature
    Ambient Pressure                                    298.00  K
    Ambient Absolute Humidity                            0.980  atm
    Ambient Relative Humidity                        1.030E-02  kg/kg BDA
                                                         50.00  %
    Input:          Mole fraction    CONCENTRATION OF C      GAS DENSITY
                                          kg/m**3              kg/m**3
                       0.00000            0.00000              1.15391
                       0.01400            0.00982              1.18584
                       0.02000            0.01413              1.19202

                       0.04200            0.03050              1.21362
                       0.06200            0.04636              1.23922
                       0.08100            0.06260              1.27043

                       0.10500            0.08523              1.32068
                       0.14200            0.12554              1.41554
                       0.16100            0.14863              1.46582

                       0.17600            0.16757              1.50182
                       0.18900            0.18413              1.52779
                       0.20400            0.20309              1.55079

                       0.22300            0.22698              1.57199
                       0.44300            0.55941              1.75560
                       0.55600            0.80191              1.89097

                       0.65400            1.08196              2.05548
                       0.73900            1.41017              2.25717
                       0.81200            1.79709              2.50469

                       0.87400            2.25592              2.80802
                       0.92500            2.79072              3.17553
                       0.96700            3.42418              3.62290

                       1.00000            4.14949              4.14949

     Specified Gas Properties:

        Molecular weight:
        Storage temperature:                                       17.000
        Density at storage temperature and ambient pressure:       298.00     K
        Mean heat capacity constant:                               4.1495     kg/m**3
        Mean heat capacity power:                                  3845.0
        Upper mole fraction contour:                               1.0000
        Lower mole fraction contour:                               1.00000E-03
        Height for isopleths:                                      1.00000E-04
                                                                   0.00000E+00m
```

Figure B.7. (continued)

```
Source input data points

          Initial mass in cloud:    0.00000E+00

     Time          Contaminant      Source Radius     Contaminant      Temperature        Enthalpy
                    Mass Rate                        Mass Fraction
       s              kg/s               m         kg contam/kg mix         K               J/kg
  0.00000E+00        56.000           135.30          1.56929E-03        298.00          0.00000E+00
   60230.            56.000           135.30          1.56929E-03        298.00          0.00000E+00
   60231.         0.00000E+00      0.00000E+00        1.56929E-03        298.00          0.00000E+00
   60232.         0.00000E+00      0.00000E+00        1.56929E-03        298.00          0.00000E+00

    Calculation procedure for ALPHA:  1

    Entrainment prescription for PHI:  3

    Layer thickness ratio used for average depth:     2.1500

    Air entrainment coefficient used: 0.590

    Gravity slumping velocity coefficient used: 1.150

    Isothermal calculation

    Heat transfer not included

    Water transfer not included

                *****                   CALCULATED SOURCE PARAMETERS                    *****
    Time      Gas Radius     Height        Qstar       SZ(x=L/2.)   Mole frac C     Density      Rich No.
    sec           m            m        kg/m**2/s         m                        kg/m**3
0.000000E+00   135.300   1.100000E-05  2.902896E-04    11.1516    2.654004E-03    1.15998      0.756144
  4.79323      136.413     1.77907     2.883740E-04    11.2280    2.636814E-03    1.15992      0.756144
  23.4441      146.849     7.82487     2.719845E-04    11.9374    2.490347E-03    1.15959      0.756144

  64.4041      181.434     15.3666     2.327196E-04    14.1936    2.145477E-03    1.15880      0.756144
  105.364      219.626     17.5763     2.059423E-04    16.5237    1.919935E-03    1.15829      0.756144
  146.324      256.937     17.0954     1.896346E-04    18.6339    1.793855E-03    1.15800      0.756144

  197.524      300.368     14.6524     1.789255E-04    20.8536    1.731507E-03    1.15786      0.756144
  218.004      315.190     13.4747     1.770343E-04    21.5390    1.730059E-03    1.15785      0.756144
  228.244      317.744     13.2504     1.769059E-04    21.6129    1.732304E-03    1.15786      0.756144

Source strength [kg/s] :                          56.000    Equivalent Primary source radius [m] :        135.30
Equivalent Primary source length [m] :            270.60    Equivalent Primary source half-width [m] :    106.26

Secondary source concentration [kg/m**3] :   1.18555E-03   Secondary source SZ [m] :                     21.613

Contaminant flux rate:    1.76556E-04


Secondary source mass fractions... contaminant:  1.023911E-03    air:    0.98879
         Enthalpy:    0.00000E+00        Density:      1.1579

Secondary source length [m] :                     635.49    Secondary source half-width [m] :             249.56

Distance     Mole    Concentration Density  Temperature   Half        Sz          Sy      Width at z=  0.00 m to:
           Fraction                                      Width                           1.000E-02moleZ 0.100    moleZ
  (m)                 (kg/m**3)  (kg/m**3)     (K)        (m)        (m)         (m)          (m)       (m)
  880.     1.732E-03  1.186E-03   1.16         298.       250.       21.6     0.000E+00     250.       250.
  884.     1.729E-03  1.183E-03   1.16         298.       237.       21.6       15.4        263.       248.
  894.     1.719E-03  1.176E-03   1.16         298.       226.       21.5       29.8        276.       248.

  914.     1.704E-03  1.166E-03   1.16         298.       215.       21.4       46.9        294.       249.
  949.     1.678E-03  1.148E-03   1.16         298.       204.       21.2       67.4        317.       252.
1.009E+03  1.631E-03  1.116E-03   1.16         298.       192.       20.9       93.2        348.       258.

1.089E+03  1.569E-03  1.073E-03   1.16         298.       184.       20.7       120.        384.       265.
1.169E+03  1.507E-03  1.030E-03   1.16         298.       179.       20.5       143.        415.       270.
1.249E+03  1.444E-03  9.878E-04   1.16         298.       175.       20.4       164.        443.       274.
```

Figure B.7. (continued)

```
 1.329E+03  1.383E-03  9.457E-04   1.16      298.      173.      20.4     183.     469.     277.
 1.409E+03  1.323E-03  9.045E-04   1.16      298.      171.      20.5     201.     493.     277.
 1.489E+03  1.265E-03  8.644E-04   1.16      298.      170.      20.6     218.     516.     275.

 1.569E+03  1.208E-03  8.255E-04   1.16      298.      169.      20.8     234.     538.     270.
 1.644E+03  1.157E-03  7.904E-04   1.16      298.      168.      21.0     248.     557.     263.
 1.724E+03  1.104E-03  7.543E-04   1.16      298.      168.      21.2     264.     576.     251.

 1.804E+03  1.054E-03  7.198E-04   1.16      298.      167.      21.5     278.     595.     231.
 1.884E+03  1.005E-03  6.868E-04   1.16      298.      167.      21.8     293.     612.     189.
 1.964E+03  9.595E-04  6.555E-04   1.16      298.      167.      22.2     307.     628.

 2.044E+03  9.160E-04  6.256E-04   1.16      298.      167.      22.5     321.     644.
 2.124E+03  8.746E-04  5.973E-04   1.16      298.      167.      22.9     334.     659.
 2.204E+03  8.353E-04  5.704E-04   1.16      298.      166.      23.3     347.     673.

 2.284E+03  7.981E-04  5.450E-04   1.16      298.      166.      23.8     360.     686.
 2.364E+03  7.629E-04  5.209E-04   1.16      298.      166.      24.2     373.     698.
 2.444E+03  7.296E-04  4.981E-04   1.16      298.      166.      24.7     386.     710.

 2.524E+03  6.982E-04  4.766E-04   1.16      298.      166.      25.2     398.     721.
 2.604E+03  6.684E-04  4.563E-04   1.16      298.      165.      25.7     411.     732.
 2.684E+03  6.403E-04  4.370E-04   1.16      298.      165.      26.2     423.     742.

 2.764E+03  6.137E-04  4.188E-04   1.16      298.      165.      26.7     435.     751.
 2.844E+03  5.885E-04  4.017E-04   1.16      298.      164.      27.2     447.     760.
 2.924E+03  5.647E-04  3.854E-04   1.16      298.      164.      27.8     459.     768.

 3.004E+03  5.422E-04  3.700E-04   1.16      298.      164.      28.3     470.     776.
 3.084E+03  5.210E-04  3.555E-04   1.16      298.      163.      28.9     482.     783.
 3.164E+03  5.008E-04  3.417E-04   1.16      298.      163.      29.5     493.     789.

 3.244E+03  4.817E-04  3.287E-04   1.15      298.      162.      30.1     505.     795.
 3.324E+03  4.637E-04  3.164E-04   1.15      298.      162.      30.6     516.     801.
 3.404E+03  4.465E-04  3.047E-04   1.15      298.      161.      31.2     527.     806.

 3.484E+03  4.303E-04  2.936E-04   1.15      298.      160.      31.8     538.     811.
 3.564E+03  4.149E-04  2.830E-04   1.15      298.      160.      32.4     549.     815.
 3.644E+03  4.002E-04  2.730E-04   1.15      298.      159.      33.1     560.     819.

 3.724E+03  3.863E-04  2.636E-04   1.15      298.      158.      33.7     571.     822.
 3.804E+03  3.731E-04  2.545E-04   1.15      298.      157.      34.3     582.     825.
 3.884E+03  3.606E-04  2.460E-04   1.15      298.      156.      34.9     593.     828.

 3.964E+03  3.486E-04  2.378E-04   1.15      298.      155.      35.6     603.     830.
 4.044E+03  3.372E-04  2.300E-04   1.15      298.      155.      36.2     614.     831.
 4.124E+03  3.264E-04  2.226E-04   1.15      298.      154.      36.8     624.     833.

 4.204E+03  3.161E-04  2.156E-04   1.15      298.      153.      37.5     635.     834.
 4.284E+03  3.062E-04  2.089E-04   1.15      298.      152.      38.1     645.     834.
 4.364E+03  2.968E-04  2.024E-04   1.15      298.      151.      38.8     655.     834.

 4.444E+03  2.878E-04  1.963E-04   1.15      298.      150.      39.4     665.     834.
 4.524E+03  2.792E-04  1.904E-04   1.15      298.      148.      40.1     676.     833.
 4.604E+03  2.710E-04  1.849E-04   1.15      298.      147.      40.7     686.     832.

 4.684E+03  2.632E-04  1.795E-04   1.15      298.      146.      41.4     696.     831.
 4.764E+03  2.557E-04  1.744E-04   1.15      298.      145.      42.1     706.     829.
 4.844E+03  2.485E-04  1.695E-04   1.15      298.      144.      42.7     716.     827.

 4.924E+03  2.416E-04  1.648E-04   1.15      298.      143.      43.4     726.     824.
 5.004E+03  2.350E-04  1.603E-04   1.15      298.      141.      44.1     735.     821.
 5.084E+03  2.287E-04  1.559E-04   1.15      298.      140.      44.8     745.     818.

 5.164E+03  2.226E-04  1.518E-04   1.15      298.      139.      45.4     755.     814.
 5.244E+03  2.168E-04  1.478E-04   1.15      298.      138.      46.1     765.     810.
 5.324E+03  2.112E-04  1.440E-04   1.15      298.      136.      46.8     774.     806.

 5.404E+03  2.058E-04  1.403E-04   1.15      298.      135.      47.5     784.     801.
 5.484E+03  2.006E-04  1.368E-04   1.15      298.      133.      48.1     794.     796.
 5.564E+03  1.957E-04  1.334E-04   1.15      298.      132.      48.8     803.     790.

 5.644E+03  1.909E-04  1.301E-04   1.15      298.      131.      49.5     813.     784.
 5.724E+03  1.863E-04  1.270E-04   1.15      298.      129.      50.2     822.     778.
 5.804E+03  1.818E-04  1.240E-04   1.15      298.      128.      50.9     831.     771.

 5.884E+03  1.775E-04  1.210E-04   1.15      298.      126.      51.6     841.     763.
 5.964E+03  1.734E-04  1.182E-04   1.15      298.      125.      52.3     850.     756.
 6.044E+03  1.694E-04  1.155E-04   1.15      298.      123.      53.0     859.     747.
```

Figure B.7. (continued)

```
6.124E+03  1.656E-04  1.129E-04  1.15   298.   122.    53.7   869.        739.
6.204E+03  1.619E-04  1.104E-04  1.15   298.   120.    54.3   878.        730.
6.284E+03  1.583E-04  1.079E-04  1.15   298.   119.    55.0   887.        720.

6.364E+03  1.548E-04  1.056E-04  1.15   298.   117.    55.7   896.        710.
6.444E+03  1.515E-04  1.033E-04  1.15   298.   116.    56.4   905.        699.
6.524E+03  1.483E-04  1.011E-04  1.15   298.   114.    57.1   914.        688.

6.604E+03  1.452E-04  9.896E-05  1.15   298.   112.    57.8   923.        676.
6.684E+03  1.421E-04  9.690E-05  1.15   298.   111.    58.5   932.        664.
6.764E+03  1.392E-04  9.491E-05  1.15   298.   109.    59.2   941.        651.

6.844E+03  1.364E-04  9.298E-05  1.15   298.   107.    59.9   950.        637.
6.924E+03  1.336E-04  9.111E-05  1.15   298.   106.    60.6   959.        622.
7.004E+03  1.310E-04  8.929E-05  1.15   298.   104.    61.3   968.        607.

7.084E+03  1.284E-04  8.754E-05  1.15   298.   102.    62.0   977.        591.
7.164E+03  1.259E-04  8.584E-05  1.15   298.   101.    62.7   986.        574.
7.244E+03  1.235E-04  8.419E-05  1.15   298.    99.0   63.4   994.        556.

7.324E+03  1.211E-04  8.259E-05  1.15   298.    97.3   64.1   1.003E+03   537.
7.404E+03  1.189E-04  8.103E-05  1.15   298.    95.6   64.8   1.012E+03   516.
7.484E+03  1.166E-04  7.952E-05  1.15   298.    93.8   65.5   1.021E+03   494.

7.564E+03  1.145E-04  7.806E-05  1.15   298.    92.1   66.2   1.029E+03   471.
7.644E+03  1.124E-04  7.664E-05  1.15   298.    90.3   66.9   1.038E+03   445.
7.724E+03  1.104E-04  7.526E-05  1.15   298.    88.5   67.6   1.046E+03   418.

7.804E+03  1.084E-04  7.391E-05  1.15   298.    86.7   68.3   1.055E+03   387.
7.884E+03  1.065E-04  7.261E-05  1.15   298.    84.9   69.0   1.064E+03   352.
7.964E+03  1.046E-04  7.134E-05  1.15   298.    83.1   69.7   1.072E+03   312.

8.044E+03  1.028E-04  7.010E-05  1.15   298.    81.3   70.5   1.081E+03   262.
8.124E+03  1.011E-04  6.890E-05  1.15   298.    79.5   71.2   1.089E+03   192.
8.204E+03  9.936E-05  6.773E-05  1.15   298.    77.6   71.9   1.098E+03

8.284E+03  9.769E-05  6.660E-05  1.15   298.    75.8   72.6   1.106E+03
8.364E+03  9.607E-05  6.549E-05  1.15   298.    74.0   73.3   1.114E+03
8.444E+03  9.448E-05  6.441E-05  1.15   298.    72.1   74.0   1.123E+03

8.524E+03  9.294E-05  6.336E-05  1.15   298.    70.2   74.7   1.131E+03
8.604E+03  9.144E-05  6.234E-05  1.15   298.    68.3   75.4   1.140E+03
8.684E+03  8.998E-05  6.134E-05  1.15   298.    66.5   76.1   1.148E+03

8.759E+03  8.864E-05  6.043E-05  1.15   298.    64.7   76.8   1.156E+03
8.839E+03  8.725E-05  5.948E-05  1.15   298.    62.8   77.5   1.164E+03
8.919E+03  8.589E-05  5.855E-05  1.15   298.    60.9   78.2   1.172E+03

8.999E+03  8.457E-05  5.765E-05  1.15   298.    58.9   78.9   1.180E+03
9.079E+03  8.327E-05  5.677E-05  1.15   298.    57.0   79.6   1.189E+03
9.159E+03  8.201E-05  5.591E-05  1.15   298.    55.1   80.3   1.197E+03

9.239E+03  8.078E-05  5.507E-05  1.15   298.    53.2   81.0   1.205E+03
9.319E+03  7.958E-05  5.425E-05  1.15   298.    51.2   81.7   1.213E+03
9.399E+03  7.841E-05  5.345E-05  1.15   298.    49.3   82.4   1.221E+03

9.479E+03  7.726E-05  5.267E-05  1.15   298.    47.3   83.1   1.229E+03
9.559E+03  7.614E-05  5.190E-05  1.15   298.    45.3   83.8   1.237E+03
9.639E+03  7.505E-05  5.116E-05  1.15   298.    43.4   84.5   1.245E+03

9.719E+03  7.398E-05  5.043E-05  1.15   298.    41.4   85.2   1.253E+03
9.799E+03  7.293E-05  4.972E-05  1.15   298.    39.4   85.9   1.261E+03
9.879E+03  7.191E-05  4.902E-05  1.15   298.    37.4   86.6   1.269E+03

9.959E+03  7.091E-05  4.834E-05  1.15   298.    35.4   87.3   1.277E+03
1.004E+04  6.994E-05  4.767E-05  1.15   298.    33.4   88.0   1.285E+03
1.011E+04  6.904E-05  4.706E-05  1.15   298.    31.5   88.7   1.293E+03

1.018E+04  6.822E-05  4.650E-05  1.15   298.    29.8   89.3   1.300E+03
1.025E+04  6.747E-05  4.600E-05  1.15   298.    28.2   89.9   1.306E+03
1.031E+04  6.680E-05  4.553E-05  1.15   298.    26.6   90.4   1.312E+03

1.036E+04  6.618E-05  4.512E-05  1.15   298.    25.3   90.9   1.318E+03
1.042E+04  6.558E-05  4.470E-05  1.15   298.    23.9   91.4   1.323E+03
1.047E+04  6.504E-05  4.434E-05  1.15   298.    22.6   91.8   1.328E+03

1.052E+04  6.451E-05  4.397E-05  1.15   298.    21.3   92.3   1.333E+03
1.056E+04  6.403E-05  4.365E-05  1.15   298.    20.2   92.7   1.337E+03
1.060E+04  6.362E-05  4.336E-05  1.15   298.    19.2   93.0   1.341E+03
```

Figure B.7. (continued)

```
1.064E+04  6.320E-05  4.308E-05  1.15    298.    18.1       93.4    1.345E+03
1.068E+04  6.279E-05  4.280E-05  1.15    298.    17.1       93.7    1.349E+03
1.072E+04  6.244E-05  4.256E-05  1.15    298.    16.2       94.0    1.352E+03

1.075E+04  6.209E-05  4.232E-05  1.15    298.    15.3       94.3    1.356E+03
1.078E+04  6.179E-05  4.212E-05  1.15    298.    14.6       94.6    1.359E+03
1.081E+04  6.150E-05  4.192E-05  1.15    298.    13.8       94.9    1.362E+03

1.084E+04  6.120E-05  4.172E-05  1.15    298.    13.0       95.1    1.365E+03
1.087E+04  6.096E-05  4.155E-05  1.15    298.    12.4       95.3    1.367E+03
1.089E+04  6.072E-05  4.139E-05  1.15    298.    11.7       95.6    1.369E+03

1.092E+04  6.048E-05  4.123E-05  1.15    298.    11.1       95.8    1.372E+03
1.094E+04  6.029E-05  4.110E-05  1.15    298.    10.6       96.0    1.374E+03
1.096E+04  6.010E-05  4.097E-05  1.15    298.    10.1       96.1    1.376E+03

1.098E+04  5.991E-05  4.084E-05  1.15    298.    9.55       96.3    1.378E+03
1.100E+04  5.972E-05  4.071E-05  1.15    298.    9.03       96.5    1.380E+03
1.101E+04  5.958E-05  4.061E-05  1.15    298.    8.64       96.6    1.381E+03

1.103E+04  5.944E-05  4.052E-05  1.15    298.    8.28       96.8    1.382E+03
1.104E+04  5.930E-05  4.042E-05  1.15    298.    7.87       96.9    1.384E+03
1.106E+04  5.916E-05  4.033E-05  1.15    298.    7.48       97.0    1.385E+03

1.107E+04  5.903E-05  4.023E-05  1.15    298.    7.10       97.2    1.387E+03
1.109E+04  5.889E-05  4.014E-05  1.15    298.    6.71       97.3    1.388E+03
1.110E+04  5.880E-05  4.008E-05  1.15    298.    6.45       97.4    1.389E+03

1.111E+04  5.871E-05  4.002E-05  1.15    298.    6.19       97.5    1.390E+03
1.112E+04  5.861E-05  3.995E-05  1.15    298.    5.93       97.5    1.391E+03
1.113E+04  5.852E-05  3.989E-05  1.15    298.    5.68       97.6    1.392E+03

1.114E+04  5.843E-05  3.983E-05  1.15    298.    5.42       97.7    1.393E+03
1.115E+04  5.834E-05  3.977E-05  1.15    298.    5.16       97.8    1.394E+03
1.116E+04  5.825E-05  3.971E-05  1.15    298.    4.90       97.9    1.395E+03

1.117E+04  5.816E-05  3.965E-05  1.15    298.    4.64       98.0    1.396E+03
1.118E+04  5.807E-05  3.959E-05  1.15    298.    4.38       98.1    1.397E+03
1.119E+04  5.798E-05  3.952E-05  1.15    298.    4.12       98.2    1.398E+03

1.119E+04  5.794E-05  3.949E-05  1.15    298.    3.99       98.2    1.398E+03
1.120E+04  5.785E-05  3.943E-05  1.15    298.    3.73       98.3    1.399E+03
1.121E+04  5.781E-05  3.940E-05  1.15    298.    3.60       98.3    1.400E+03

1.122E+04  5.772E-05  3.934E-05  1.15    298.    3.35       98.4    1.401E+03
1.122E+04  5.767E-05  3.931E-05  1.15    298.    3.22       98.5    1.401E+03
1.123E+04  5.758E-05  3.925E-05  1.15    298.    2.96       98.6    1.402E+03

1.124E+04  5.754E-05  3.922E-05  1.15    298.    2.83       98.6    1.403E+03
1.125E+04  5.745E-05  3.916E-05  1.15    298.    2.57       98.7    1.404E+03
1.125E+04  5.741E-05  3.913E-05  1.15    298.    2.44       98.7    1.404E+03

1.126E+04  5.732E-05  3.907E-05  1.15    298.    2.18       98.8    1.405E+03
1.127E+04  5.728E-05  3.904E-05  1.15    298.    2.05       98.9    1.406E+03
1.127E+04  5.723E-05  3.901E-05  1.15    298.    1.92       98.9    1.406E+03

1.128E+04  5.719E-05  3.898E-05  1.15    298.    1.79       99.0    1.407E+03
1.128E+04  5.714E-05  3.895E-05  1.15    298.    1.66       99.0    1.407E+03
1.129E+04  5.710E-05  3.892E-05  1.15    298.    1.53       99.0    1.407E+03

1.129E+04  5.706E-05  3.889E-05  1.15    298.    1.40       99.1    1.408E+03
1.130E+04  5.701E-05  3.886E-05  1.15    298.    1.27       99.1    1.408E+03
1.130E+04  5.697E-05  3.883E-05  1.15    298.    1.14       99.2    1.409E+03

1.131E+04  5.693E-05  3.880E-05  1.15    298.    1.01       99.2    1.409E+03
1.131E+04  5.688E-05  3.877E-05  1.15    298.    0.880      99.3    1.410E+03
1.132E+04  5.684E-05  3.875E-05  1.15    298.    0.750      99.3    1.410E+03

1.132E+04  5.680E-05  3.872E-05  1.15    298.    0.620      99.3    1.411E+03
1.133E+04  5.675E-05  3.869E-05  1.15    298.    0.490      99.4    1.411E+03
1.133E+04  5.671E-05  3.866E-05  1.15    298.    0.360      99.4    1.412E+03

1.134E+04  5.667E-05  3.863E-05  1.15    298.    0.230      99.5    1.412E+03
1.134E+04  5.662E-05  3.860E-05  1.15    298.    0.100      99.5    1.413E+03
1.135E+04  5.658E-05  3.857E-05  1.15    298.    0.000E+00  99.6    1.412E+03

1.142E+04  5.570E-05  3.797E-05  1.15    298.    0.000E+00  101.    1.420E+03
1.150E+04  5.474E-05  3.732E-05  1.15    298.    0.000E+00  102.    1.427E+03
1.158E+04  5.381E-05  3.668E-05  1.15    298.    0.000E+00  103.    1.435E+03
```

Figure B.7. (continued)

```
1.166E+04  5.290E-05  3.606E-05  1.15      298.     0.000E+00  104.      1.443E+03
1.174E+04  5.202E-05  3.546E-05  1.15      298.     0.000E+00  105.      1.450E+03
1.182E+04  5.115E-05  3.487E-05  1.15      298.     0.000E+00  106.      1.458E+03

1.190E+04  5.031E-05  3.430E-05  1.15      298.     0.000E+00  107.      1.466E+03
1.194E+04  4.995E-05  3.405E-05  1.15      298.     0.000E+00  107.      1.469E+03

For the UFL of   0.10000    mole percent, and the LFL of  1.00000E-02 mole percent:
The mass of contaminant between the UFL and LFL is:  40143.    kg.
The mass of contaminant above the LFL is:   42885.   kg.
```

Figure B.7. (concluded)

Burro Test LNG Releases

In 1980, the U.S. Department of Energy sponsored at China Lake, California, the BURRO series of LNG releases (Koopman, et al., 1982). The Burro tests were ground-level releases; their simulation does not require use of the JETPLU model. Burro 9 (Table B.3) was modeled here, both as a steady-state and transient (time-limited) release, using the DEGADIS model with interactive input via DEGADISIN.

Table B.3. Burro 9 Test Conditions

Source rate	130.0 kg/s
Source radius	22.06 m
Wind speed	6.5 m/s at 8.0 m
Atmospheric stability	C (Pasquill)
Monin-Obukhov length	-140. m
Surface roughness	2.05×10^{-4} m
Air temperature	33.4 C
Atmospheric humidity	12.5 %
Surface temperature	310 K

The source radius assumes LNG release onto water, with a constant evaporative flux of 0.085 kg/m^2 s. The input procedures for simulation of the transient release (RUN_NAME=BURRO9) and the steady-state release (RUN_NAME=BURRO9S) are very similar. Therefore, only the specification of the surface rate and extent have been included for the transient release. In the following line-by-line description of the input procedure:

(*) A line terminator (normally a carriage return) must end every line entered by the user.

(*) The file name specification RUN_NAME must satisfy system restrictions.

(*) When DEGADISIN requests the user to choose an option, responses are a single character (capital or lower case). The default responses are denoted by a capital letter inside angle brackets (e.g. <N>). When applicable, a menu of responses is included inside parentheses.

(*) For numerical responses, a comma, space, tab, or line terminator (carriage return) may separate the numbers.

(*) When a file is used as input (i.e. for the density or transient source input), DEGADISIN reads the same information from the file which would be entered at the terminal in the same order and in the same format.

(This page is intentionally left blank.)

Notes on Steady-State Simulation of BURRO9

① Begin the input procedure by execution of DEGADISIN.

② The file name specification must follow system restrictions. The DEGADIS model uses this file name along with various file extensions for input and output.

③ The Title Block is used to carry any desired comments such as information on the specification of certain parameters.

④ The wind field parameters include the wind velocity (m/s) at a specified height (m) and the surface roughness (m).

⑤ The Pasquill stability class is used to generate estimates of other atmospheric parameters which follow.

⑥ The averaging time is used to determine the value of δ_y (DELTAY) in the lateral dispersion coefficient specification. Changes to the values of δ_y are calculated assuming that the effect of averaging time only influences the lateral plume meander of a steady-state release.

⑦ The current settings of pertinent atmospheric parameters are displayed in this list. If any of these are to be changed, the first letter of the parameter to be changed is entered. Note that the default--indicated by <N>--is No for no changes.

⑧ The Monin-Obukhov length (Length in the list) is to be changed, so L is entered in response to the prompt.

⑨ The list is redisplayed to verify the change and to request any further changes. The (default) response of No causes the program to go to the next question.

⑩ The ambient temperature and pressure are entered.

⑪ DEGADISIN calculates the ambient air density for the given input parameters.

⑫ If the release is isothermal, respond "Y". A positive response causes DEGADISIN to ask for a list of concentration, density, and mole fraction points for the gas mixture. The default response is negative.

⑬ If the release is simulated as adiabatic, the default negative response is chosen. For inclusion of heat transfer effects, the surface temperature and the method of calculating the heat transfer coefficient must be specified.

⑭ Water transfer to the source blanket (if present) can be included in the calculation.

```
①  RUN SYSSDEGADIS:DEGADISIN

                     DEnse GAs DISpersion Model input module.
②  Enter the simulation name : [DIR]RUNNAME BURRO9S
        INPUT MODULE -- DEGADIS MODEL

        ************************************
③      Enter Title Block -- up to 4 lines of 80 characters
        To stop, type "//"
    Steady-state simulation of BURRO 9
    //
④      ENTER WIND PARAMETERS -- U0 (m/s), Z0 (m), and ZR(m)
        U0 -- Wind velocity at reference height Z0
        ZR -- Surface Roughness
    6.5,8.,2.05E-4

⑤      Enter the Pasquill stability class: (A,B,C,D,E,F) <D> C
⑥  Enter the averaging time (s) for estimating DELTAY: 0.
⑦
    The values for the atmospheric parameters are set as follows:
    DELTAY:
                                 0.1046
    BETAY:
                                 0.9000
    Monin-Obukhov length:       -9.3344 m
    Sigma X Coefficient:         0.0200
    Sigma X Power:               1.2200
    Sigma X Minimum Distance:  130.0000 m
⑧  Do you wish to change any of these?
    (No,Deltay,Betay,Length,Coefficient,Power,Minimum) <N> L
    Note: For infinity, RML = 0.0
    Enter the desired Monin-Obukhov length: (m) -140.

⑨  The values for the atmospheric parameters are set as follows:
    DELTAY:
                                 0.1046
    BETAY:
                                 0.9000
    Monin-Obukhov length:     -140.0000 m
    Sigma X Coefficient:         0.0200
    Sigma X Power:               1.2200
    Sigma X Minimum Distance:  130.0000 m
    Do you wish to change any of these?
    (No,Deltay,Betay,Length,Coefficient,Power,Minimum) <N>

⑩  Enter the ambient temperature(K) and pressure(atm): 305.88,0.94

    The ambient humidity can be entered as Relative or Absolute.
    Enter either R or A <R or a>:
    Enter the relative humidity (%): 12.5

⑪  Ambient Air density is    1.081886     kg/m**3

⑫  Is this an Isothermal spill? <y or N>

⑬  Is heat transfer to be included in the calculations <y or N> Y
    Enter the surface temperature [=] K : 310.
    Do you want to use the built in correlation, the LLNL correlation, or
    enter a particular value?
    (Corr,LLNLcorr,Value)  <C>

⑭  Is water transfer to be included in the source <y or N>
```

⑮ Enter the three-letter designation of the diffusing gas. The properties of LNG as methane, LPG as propane, and NH_3 (ammonia) are included among others.

⑯ A list of the properties for the specified gas (if available) is given. If any of the parameters are to be changed, the first letter of the parameter to be changed in the list is given to the prompt. Here, the level at which the flammability contours are calculated is changed from 0.5 m to 1.0 m.

⑰ The gas property list is displayed again. The default response is no change.

⑱ The lowest concentration of interest is the concentration at which the calculations are stopped.

⑲ The BURRO9 case is for a release of "pure" (undiluted) LNG. For diluted releases, DEGADIS requires the mass fraction of the contaminant and the mixture temperature.

⑳ If a steady-state release is to be simulated, type "Y" to the prompt. For a steady simulation, the steady-state mass evolution rate (kg/s) and primary source extent (m) are required.

㉑ A note about the numerical parameter files is included. These files contain various constant values used in the programs to which the user has access without recompiling the programs. Access is granted as a convenience.

㉒ DEGADISIN will generate a command procedure suitable for running the model under VMS.

㉓ If so desired, DEGADISIN will initiate the command procedure under VMS. If not, the program returns to the operating system.

```
⑮  Enter the code name of the diffusing species: LNG
⑯  The characteristics for the gas are set as follows:
    Molecular weight:                                      16.04
    Storage temperature [K]:                               111.70
    Density at storage temperature, PAMB  [kg/m**3]:       1.6845
    Mean Heat capacity constant                           5.60000E-08
    Mean Heat capacity power                               5.0000
    Upper Flammability Limit [mole frac]                   0.15000
    Lower Flammability Limit [mole frac]                   5.00000E-02
    Height of Flammability Limit [m]                       0.50000
    Do you wish to change any of these? (No,Mole,Temp,Den,Heat,Power,Upper,Lower,Z)
    <N> Z
    Enter the desired Height for the flammable limit calculations: 1.
⑰  The characteristics for the gas are set as follows:
    Molecular weight:                                      16.04
    Storage temperature [K]:                               111.70
    Density at storage temperature, PAMB  [kg/m**3]:       1.6845
    Mean Heat capacity constant                           5.60000E-08
    Mean Heat capacity power                               5.0000
    Upper Flammability Limit [mole frac]                   0.15000
    Lower Flammability Limit [mole frac]                   5.00000E-02
    Height of Flammability Limit [m]                       1.0000
    Do you wish to change any of these? (No,Mole,Temp,Den,Heat,Power,Upper,Lower,Z)
    <N>
⑱      The suggested LOWEST CONCENTRATION OF INTEREST (gas_lfl/2.)
        is   1.50191E-02 kg/m**3.  Enter the desired value: 0.015

    Specification of source parameters.

⑲  Is this a release of pure (P) or diluted (d) material specified above? <P or d>

⑳  Is this a Steady state simulation? <y or N> Y

    Enter the desired evolution rate [=] kg/sec : 130.
    Enter the desired source radius [=] m : 22.06
㉑  In addition to the information just obtained, DEGADIS
    requires a series of numerical parameter files which use
    the same name as [DIR]RUNNAME given above.

       For convenience, example parameter files are included for
    each step.  They are:
            EXAMPLE.ER1 and
            EXAMPLE.ER2
     Note that each of these files can be edited during the course of the
    simulation if a parameter proves to be out of specification.
㉒  Do you want a command file to be generated to execute the procedure? <Y or n>
    The command file will be generated under the file name:
            BURRO8S.com
㉓  Do you wish to initiate this procedure?  <y or N>
    $
```

Notes on Transient Simulation of BURRO9

Beginning with the specification of the source rate and extent, the responses to all of the previous questions except the simulation name (RUN_NAME) are the same for the steady-state case and are not repeated.

⑲ The BURRO9 case is for a release of "pure" (undiluted) LNG. For diluted releases, DEGADIS requires the mass fraction of the contaminant and the mixture temperature.

⑳ The default response is for a transient release.

㉑ An initial mass of gas can be specified over the source. This can be used to model aboveground releases such as the Thorney Island Trials.

㉒ The transient source description consists of ordered triples of time, evolution rate, and source radius for pure sources. For diluted sources, values of evolution rate, source radius, contaminant mass fraction, and source temperature must be specified as functions of time.

㉓ An input file can be used to enter the data triples to avoid typing errors or to use as output from another model such as a liquid spreading model. The file format is the same as the terminal entry format.

㉔ The first item is the number of triples used in the description followed by the triples with the last two values showing no gas present.

㉕ A note about the numerical parameter files is included. These files contain various constant values used in the programs to which the user has access without recompiling the programs. Access is granted as a convenience.

㉖ DEGADISIN will generate a command procedure suitable for running the model under VMS.

㉗ If so desired, DEGADISIN will initiate the procedure under VMS. If not, the program returns to the operating system.

```
⑲  Specification of source parameters.

⑳  Is this a release of pure (P) or diluted (d) material specified above? <P or d>

㉑  Is this a Steady state simulation? <y or N>

    Enter the initial mass of pure gas over the source. (kg)
    (Positive or zero): 0.
㉒
                   Source Description

        The description of the primary source mass
        evolution rate E and radius R1 for a transient
        release is input by ordered triples as follows:

          first point        -- t=0, E(t=0), R1(t=0) (initial, nonzero values)
          second point       -- t=t1, E(t=t1), R1(t=t1)
               .
               .
               .

          last nonzero point -- t=TEND, E(t=TEND), R1(t=TEND)
          next to last point -- t=TEND+1., E=0., R1=0.
          last point         -- t=TEND+2., E=0., R1=0.

        Note: the final time (TEND) is the last time when E and R1 are non-zero.
㉓  Do you have an input file for the Source Description? [y or N]

        Enter the number of triples (max= 30) starting with t=0. and ending
        with t=TEND+2. for the source description: 4
㉔
        Enter TIME (sec), EVOLUTION RATE (kg/s), and POOL RADIUS (m)
    0.,130.,22.06
    80.,130.,22.06
    81.,0.,0.
    82.,0.,0.

㉕  In addition to the information just obtained, DEGADIS
    requires a series of numerical parameter files which use
    the same name as [DIR]RUNNAME given above.

       For convenience, example parameter files are included for
    each step.  They are:
            EXAMPLE.ER1,
            EXAMPLE.ER2, and
            EXAMPLE.ER3
     Note that each of these files can be edited during the course of the
    simulation if a parameter proves to be out of specification.

㉖  Do you want a command file to be generated to execute the procedure? <Y or n>
    The command file will be generated under the file name:
            BURRO9.com

㉗  Do you wish to initiate this procedure?  <y or N>
    $
```

The INP files for BURRO9S and BURRO9 are shown in Figures B.8 and B.9. If necessary, the user may edit the INP file before beginning the simulation.

Example Simulation Output

BURRO9.LIS and BURRO9S.LIS contain the output listing for the transient and steady-state releases, respectively. A discussion of the steady-state and transient simulation listings follows. Because of the similarities between the steady-state and transient simulation listings, the first portion of the transient simulation is not included.

```
Steady-state simulation of BURRO 9


  6.500000       8.000000      2.0500000E-04
   3
 0.0000000E+00  0.0000000E+00
 0.1046050      0.9000000      -140.0000
 2.0000000E-02   1.220000       130.0000
  305.8800      0.9400000      4.3809577E-03   12.50000
   0   310.0000
   1  0.0000000E+00
   0  0.0000000E+00
LNG
  16.04000       111.7000      1.684480
 5.6000000E-08   5.000000
 0.1500000      5.0000000E-02  1.000000
 1.5000000E-02
 0.0000000E+00
   4
 0.0000000E+00  130.0000       22.06000
  6023.000      130.0000       22.06000       1.000000       111.7000       1.000000
  6024.000     0.0000000E+00  0.0000000E+00   1.000000       111.7000       1.000000
  6025.000     0.0000000E+00  0.0000000E+00   1.000000       111.7000       1.000000
F F F F T F                                   1.000000       111.7000       1.000000
9-JUN-1989 15:55:42.38
  130.0000      44.12000       17.32588
```

Figure B.8. BURRO9S.INP Listing.

```
Time-limited simulation of Burro 9


  6.500000       8.000000      2.0500000E-04
   3
 0.0000000E+00  0.0000000E+00
 0.1046050      0.9000000      -140.0000
 2.0000000E-02   1.220000       130.0000
  305.8800      0.9400000      4.3809577E-03   12.50000
   0   310.0000
   1  0.0000000E+00
   0  0.0000000E+00
LNG
  16.04000       111.7000      1.684480
 5.6000000E-08   5.000000
 0.1500000      5.0000000E-02  1.000000
 1.5000000E-02
 0.0000000E+00
   4
 0.0000000E+00  130.0000       22.06000
  80.00000      130.0000       22.06000       1.000000       111.7000       1.000000
  81.00000     0.0000000E+00  0.0000000E+00   1.000000       111.7000       1.000000
  82.00000     0.0000000E+00  0.0000000E+00   1.000000       111.7000       1.000000
F F F F F F                                   1.000000       111.7000       1.000000
9-JUN-1989 16:02:24.47
```

Figure B.9. BURRO9.INP Listing.

Notes on Steady-State Simulation of BURRO9

① The date and time DEGADISIN was run are included.

② The input information gathered by DEGADISIN is repeated to assist in documentation of the simulations. Included here are the Title Block and the atmospheric conditions.

```
   ********************    U O A _ D E G A D I S   M O D E L   O U T P U T  - -  V E R S I O N  2.1      **********
   ***************                 ***************  6-SEP-1989 16:05:07.54  ***************                   *****
①  Data input on                    9-JUN-1989 15:55:42.38
   Source program run on            6-SEP-1989 16:05:07.54

②         TITLE BLOCK

   Steady-state simulation of BURRO 9


      Wind velocity at reference height                         6.50  m/s
      Reference height                                          8.00  m

      Surface roughness length                             2.050E-04  m

      Pasquill Stability class                                  C

      Monin-Obukhov length                                     -140.  m
      Gaussian distribution constants
                    Specified averaging time                    0.00  s
                                       Delta                 0.10461
                                        Beta                 0.90000

      Wind velocity power law constant  Alpha                0.10638
      Friction velocity                                      0.21878  m/s

      Ambient Temperature                                     305.88  K

      Surface Temperature                                     310.00  K
      Ambient Pressure                                         0.940  atm
      Ambient Absolute Humidity                            4.381E-03  kg/kg BDA
      Ambient Relative Humidity                                12.50  %

      Adiabatic Mixing:  Mole fraction  CONCENTRATION OF C    GAS DENSITY       Enthalpy        Temperature
                                            kg/m**3             kg/m**3           J/kg               K
                           0.00000          0.00000             1.08189        0.00000E+00        305.88
                           0.00897          0.00542             1.08476        -2018.3            303.89
                           0.01786          0.01086             1.08771        -4036.6            301.93

                           0.02669          0.01636             1.09065        -6054.8            299.98
                           0.04413          0.02741             1.09652        -10091.            296.16
                           0.06131          0.03858             1.10235        -14128.            292.41

                           0.08657          0.05555             1.11110        -20183.            286.93
                           0.11126          0.07278             1.11976        -26238.            281.62
                           0.14331          0.09616             1.13126        -34311.            274.79

                                                          •

                                                          •

                                                          •

                          0.74203          0.86864             1.41242        -2.48249E+05       160.04
                          0.79281          0.98925             1.45478        -2.74487E+05       150.27
                          0.84028          1.11628             1.49833        -3.00724E+05       141.26

                          0.88805          1.26067             1.54684        -3.28980E+05       132.29
                          0.93269          1.41351             1.59719        -3.57236E+05       124.01
                          0.97738          1.58729             1.65343        -3.87511E+05       115.81

                          1.00000          1.88448             1.88448        -4.03657E+05       111.70
```

③ Continuing with the input information, the contaminant gas properties are output.

④ The specification of the mass evolution rate, source radius, contaminant mass fraction, and source temperature are output. For a steady-state release, there is no initial mass in the cloud, and the source parameters are held constant for an arbitrarily large period of time.

⑤ Finally, certain numerical parameters and calculation flags are displayed. Some of these are set in DEGADISIN while others are set in the numerical parameter files.

⑥ A summary of the calculated secondary source parameters is included. The secondary source gas radius and height are output as functions of time along with other secondary parameters including the source mass flux (Qstar), the vertical concentration distribution parameter at the downwind edge (SZ(x = L/2.)), the contaminant mole fraction (Mole frac C), the gas mixture density (Density), and the Richardson number based on the cloud spreading velocity (Rich No.).

⑦ For a steady-state release, the source calculations are terminated after the calculated parameters are no longer changing as a function of time. A summary of the steady-state secondary source is included.

⑧ The downwind portion of the calculations is included. The distance downwind of the source is given in the first column. Columns 2 through 6 contain the mole fraction, contaminant concentration, mixture density, ratio of $(\rho - \rho_a)/c_c$ (Gamma), and mixture temperature on the centerline of the gas cloud at ground level. Columns 7 through 9 contain the contour shape parameters b (Half Width), S_z, and S_y. Finally, columns 10 and 11 contain the width from the centerline to the indicated concentration levels at the indicated height. Note that the output is prematurely terminated. Output actually continues until the centerline, ground-level concentration drops below the lowest concentration of interest.

③

```
Specified Gas Properties:

  Molecular weight:                                              16.040
  Storage temperature:                                           111.70     K
  Density at storage temperature and ambient pressure:           1.6845     kg/m**3
  Mean heat capacity constant:                                  5.60000E-08
  Mean heat capacity power:                                      5.0000
  Upper mole fraction contour:                                  0.15000
  Lower mole fraction contour:                                  5.00000E-02
  Height for isopleths:                                          1.0000     m
```

④

```
Source input data points

            Initial mass in cloud:   0.00000E+00

   Time        Contaminant       Source Radius    Contaminant        Temperature        Enthalpy
                Mass Rate                         Mass Fraction
    s             kg/s                m           kg contam/kg mix        K                J/kg
 0.00000E+00     130.00            22.060            1.0000            111.70          -4.03657E+05
  6023.0         130.00            22.060            1.0000            111.70          -4.03657E+05
  6024.0       0.00000E+00       0.00000E+00         1.0000            111.70          -4.03657E+05
  6025.0       0.00000E+00       0.00000E+00         1.0000            111.70          -4.03657E+05
```

⑤

```
Calculation procedure for ALPHA:  1

Entrainment prescription for PHI:  3

Layer thickness ratio used for average depth:     2.1500

Air entrainment coefficient used: 0.590

Gravity slumping velocity coefficient used: 1.150

NON Isothermal calculation

Heat transfer calculated with correlation:  1

Water transfer not included
```

⑥

```
                  *****
                                      CALCULATED SOURCE PARAMETERS
                                                                                         *****
    Time       Gas Radius      Height           Qstar         SZ(x=L/2.)   Mole frac C    Density    Temperature    Rich No.
    sec            m             m           kg/m**2/s           m                        kg/m**3        K
 0.000000E+00   22.0600     1.100000E-05   8.329550E-02   0.498870      1.00000       1.68448      111.700      0.756144
  1.93242       22.1998     1.304959E-03   8.367377E-02   0.508692      0.999937      1.66670      112.897      0.756144
  7.25031       22.2263     1.331373E-03   8.375806E-02   0.510725      0.999935      1.66315      113.139      0.756144
```

⑦

```
Source strength [kg/s] :                           130.00
Equivalent Primary source length [m] :             44.120     Equivalent Primary source radius [m] :            22.060
                                                              Equivalent Primary source half-width [m] :        17.326
Secondary source concentration [kg/m**3] :         1.6630     Secondary source SZ [m] :
                                                                                                                0.51072
Contaminant flux rate:    8.37642E-02


Secondary source mass fractions... contaminant:  0.999884        air:   1.15782E-04
          Enthalpy:  -4.00663E+05        Density:     1.6631
Secondary source length [m] :
                                                   44.453     Secondary source half-width [m] :
                                                                                                             17.457
```

⑧

```
Distance    Mole    Concentration Density      Gamma    Temperature   Half       Sz         Sy      Width at z=  1.00 m to
          Fraction                                                    Width                      5.00    mole%   15.0    m
  (m)                (kg/m**3)  (kg/m**3)                   (K)        (m)       (m)        (m)           (m)        (m)
 22.2      1.00        1.66      1.6631      0.350        113.        17.5      0.511    0.000E+00      17.5       17.5
 22.5      0.998       1.65      1.6570      0.348        114.        16.7      0.510    0.948          18.0       17.5
 25.2      0.980       1.58      1.6061      0.332        119.        15.9      0.504    3.05           20.0       18.3
                                               •
                                               •
                                               •
```

Notes on Transient Simulation Output of BURRO9

⑥ A summary of the calculated secondary source parameters is included. The secondary source gas radius and height are output as functions of time along with other secondary parameters including the source mass flux (Qstar), the vertical concentration distribution parameter at the downwind edge (SZ(x = L/2.)), the contaminant mole fraction (Mole frac C), the gas mixture density (Density), and the Richardson number based on the cloud spreading velocity (Rich No.). Note that only a portion of the output is shown. The source calculation ends when all the primary and secondary source gas has been taken up in the atmospheric flow.

⑥

```
      *****                          CALCULATED SOURCE PARAMETERS                              *****
  Time        Gas Radius      Height          Qstar        SZ(x=L/2.)   Mole frac C    Density     Temperature    Rich No.
  sec             m              m          kg/m**2/s          m                       kg/m**3          K

0.000000E+00   22.0600     1.100000E-05  8.329550E-02  0.498870        1.00000      1.68448       111.700      0.756144
7.812500E-05   22.0600     1.108054E-05  8.329550E-02  0.498870        1.00000      1.68448       111.700      0.756144
1.562500E-04   22.0600     1.116108E-05  8.329550E-02  0.498870        1.00000      1.68448       111.700      0.756144

2.995998E-04   22.0600     1.130885E-05  8.329549E-02  0.498870        1.00000      1.68448       111.700      0.756144
4.429495E-04   22.0600     1.145663E-05  8.329549E-02  0.498870        1.00000      1.68448       111.700      0.756144
6.200244E-04   22.0600     1.163917E-05  8.329548E-02  0.498870        1.00000      1.68448       111.700      0.756144

7.970994E-04   22.0600     1.182171E-05  8.329548E-02  0.498870        1.00000      1.68448       111.700      0.756144
1.302401E-03   22.0600     1.234260E-05  8.329547E-02  0.498870        1.00000      1.68448       111.700      0.756144
5.584851E-03   22.0601     1.675684E-05  8.329537E-02  0.498870        1.00000      1.68448       111.700      0.756144

                                                        •

                                                        •

                                                        •

9.775356E-02   22.0620     1.112523E-04  8.330184E-02  0.499023       0.999998      1.68419       111.719      0.756144
0.134816       22.0631     1.488579E-04  8.330547E-02  0.499110       0.999998      1.68403       111.730      0.756144
0.184839       22.0649     1.991522E-04  8.331114E-02  0.499247       0.999996      1.68377       111.747      0.756144

0.251490       22.0676     2.652104E-04  8.331988E-02  0.499457       0.999994      1.68338       111.774      0.756144
0.355980       22.0726     3.662174E-04  8.333587E-02  0.499841       0.999991      1.68266       111.822      0.756144
0.523614       22.0821     5.206466E-04  8.336356E-02  0.500537       0.999984      1.68137       111.908      0.756144

 1.15444       22.1288     9.974017E-04  8.348547E-02  0.503776       0.999959      1.67548       112.304      0.756144
 1.47042       22.1565     1.162954E-03  8.355603E-02  0.505664       0.999948      1.67209       112.532      0.756144
 1.79110       22.1863     1.274691E-03  8.363808E-02  0.507764       0.999940      1.66834       112.786      0.756144

 2.07986       22.2139     1.324084E-03  8.371218E-02  0.509677       0.999936      1.66496       113.015      0.756144
 36.8751       22.2250     1.345084E-03  8.369677E-02  0.509883       0.999934      1.66475       113.030      0.756144
 79.9765       22.2283     1.370607E-03  8.375954E-02  0.510787       0.999933      1.66299       113.149      0.756144

 80.0499       21.8594     1.392848E-03  8.688047E-02  0.535701       0.999932      1.60926       116.927      0.756144
 80.0918       21.1203     1.375235E-03  9.024113E-02  0.548069       0.999933      1.57472       119.492      0.756144
 80.3716       15.7908     1.086961E-03  0.110060      0.530173       0.999954      1.48958       126.319      0.756144

 80.6057       11.3499     8.196970E-04  0.132618      0.486207       0.999969      1.41975       132.531      0.756144
 80.7910       7.57905     5.692027E-04  0.157312      0.408026       0.999979      1.36527       137.818      0.756144
 80.8842       5.36474     4.112889E-04  0.174600      0.334241       0.999984      1.33743       140.686      0.756144

 80.9319       3.99387     3.084351E-04  0.186437      0.273428       0.999987      1.32770       141.717      0.756144
 80.9590       3.04157     2.356705E-04  0.195637      0.223938       0.999990      1.32331       142.187      0.756144
 80.9750       2.34519     1.820257E-04  0.203107      0.183421       0.999991      1.32103       142.432      0.756144

 80.9846       1.82960     1.421910E-04  0.209143      0.150660       0.999992      1.31932       142.617      0.756144
 80.9904       1.44346     1.123626E-04  0.213940      0.124304       0.999993      1.31719       142.848      0.756144
 80.9939       1.15560     9.016800E-05  0.217575      0.102447       0.999994      1.31406       143.188      0.756144

 80.9964       0.900469    7.055920E-05  0.220619      8.393002E-02  0.999994      1.30850       143.796      0.756144
 80.9978       0.714082    5.630183E-05  0.222380      6.893466E-02  0.999995      1.30040       144.691      0.756144
 80.9986       0.582421    4.629137E-05  0.222961      5.789226E-02  0.999995      1.28998       145.861      0.756144

 80.9992       0.462480    3.725033E-05  0.222315      4.744542E-02  0.999995      1.27292       147.816      0.756144
 80.9995       0.384735    3.146655E-05  0.220537      4.044054E-02  0.999995      1.25354       150.101      0.756144
 80.9998       0.308874    2.594198E-05  0.216437      3.339515E-02  0.999995      1.22064       154.147      0.756144

 80.9999       0.251972    2.196174E-05  0.209942      2.794875E-02  0.999995      1.17617       159.974      0.756144
 81.0000       0.217901    1.971995E-05  0.203099      2.460883E-02  0.999995      1.13276       166.105      0.756144
 81.0000       0.199541    1.859155E-05  0.197788      2.278038E-02  0.999995      1.10028       171.009      0.756144

                                                        •

                                                        •

                                                        •
```

⑦ An indication of the constants used in the x-direction dispersion correction is included. The output will also indicate if the x-direction dispersion correction was not applied.

⑧ The concentration field is shown for different times after the beginning of the spill. Unless otherwise specified, DEGADIS will choose default values for the time to output the concentration field. If other times are desired, DEGADIS3 can be executed again after the appropriate changes to RUN_NAME.ER3 have been made.

⑨ The downwind portion of the calculations is included. The distance downwind of the source is given in the first column. Columns 2 through 6 contain the mole fraction, contaminant concentration, mixture density, ratio of $(\rho - \rho_a)/c_c$ (Gamma), and mixture temperature on the centerline of the gas cloud at ground level. Columns 7 through 9 contain the contour shape parameters b (Half Width), S_z, and S_y. Finally, columns 10 and 11 contain the width from the centerline to the indicated concentration levels at the indicated height. Note that the output is prematurely terminated.

```
     Sorted values for each specified time.

⑦    X-Direction correction was applied.
          Coefficient:          2.00000E-02
          Power:                 1.2200
          Minimum Distance:     130.00     m


⑧        Time after beginning of spill    12.00000      sec
⑨ Distance     Mole    Concentration Density     Gamma    Temperature   Half        Sz           Sy       Width at z=  1.00 m to:
              Fraction                                                  Width                             5.00    mole%  15.0    mole
     (m)                 (kg/m**3)  (kg/m**3)                    (K)     (m)        (m)          (m)          (m)          (m)
   31.2        1.03        1.29        1.2269   0.113        166.      14.8       0.572        5.37        22.2          19.4
   44.8       0.757       0.804        1.2680   0.232        182.      15.4       0.589        8.97        26.3          20.4
   58.6       0.500       0.425        1.1880   0.250        222.      14.0       0.602        11.3        24.5

   72.7       0.152       0.100        1.1046   0.227        281.      6.89       0.777        11.3

 For the UFL of     15.000      mole percent, and the LFL of     5.0000     mole percent:

 The mass of contaminant between the UFL and LFL is:    89.846     kg.
 The mass of contaminant above the LFL is:    641.91     kg.


        Time after beginning of spill    23.00000      sec

 Distance     Mole    Concentration Density     Gamma    Temperature   Half        Sz           Sy       Width at z=  1.00 m to:
              Fraction                                                  Width                             5.00    mole%  15.0    mole
     (m)                 (kg/m**3)  (kg/m**3)                    (K)     (m)        (m)          (m)          (m)          (m)
   29.0        1.06        1.38        1.2314   0.108        163.      14.8       0.568        4.57        21.1          18.8
   42.6       0.868        1.05        1.2857   0.195        168.      14.7       0.630        8.28        26.0          21.6
   56.4       0.759       0.842        1.3227   0.286        172.      16.0       0.662        11.2        30.6          24.4

   70.4       0.684       0.686        1.2564   0.254        190.      17.5       0.706        13.7        35.0          26.8
   84.7       0.596       0.543        1.2040   0.225        209.      18.7       0.777        16.2        38.9          28.8
   99.1       0.429       0.337        1.1457   0.189        239.      18.6       0.896        18.6        39.5          24.6

   114.       0.235       0.163        1.1145   0.201        268.      14.7        1.10        19.6        31.8
   129      6.338E-02   3.939E-02      1.0875   0.143        296.      5.46        1.47        17.7

 For the UFL of     15.000      mole percent, and the LFL of     5.0000     mole percent:

 The mass of contaminant between the UFL and LFL is:    359.47     kg.
 The mass of contaminant above the LFL is:    2105.8     kg.


        Time after beginning of spill    34.00000      sec

 Distance     Mole    Concentration Density     Gamma    Temperature   Half        Sz           Sy       Width at z=  1.00 m to:
              Fraction                                                  Width                             5.00    mole%  15.0    mole%
     (m)                 (kg/m**3)  (kg/m**3)                    (K)     (m)        (m)          (m)          (m)          (m)
   26.9        1.10        1.45        1.2190   9.445E-02    162.      15.0       0.553        3.80        20.3          18.3
   40.4       0.912        1.09        1.2781   0.180        167.      14.6       0.623        7.77        25.2          21.2
   54.1       0.778       0.871        1.3185   0.272        171.      15.7       0.657        10.7        29.9          23.9

   68.1       0.697       0.710        1.2658   0.259        187.      17.3       0.695        13.3        34.4          26.5
   82.4       0.620       0.576        1.2130   0.228        205.      18.5       0.760        15.8        38.3          28.6
   96.8       0.544       0.468        1.1756   0.200        220.      19.5       0.844        18.2        41.8          30.3
                                                              •
                                                              •
                                                              •
```

(This page is intentionally left blank.)

In addition to the output of the concentration field at specified times (from DEGADIS3), DEGADIS allows for transient releases output of concentration time histories at specified positions using DEGADIS4. DEGADIS4 can be executed interactively or in batch mode. An example DEGADIS4 interactive run and output follow.

Notes on Using DEGADIS4

① DEGADIS4 can be run interactively as demonstrated here or in batch mode by supplying the same responses demonstrated here. DEGADIS4 sends the output to RUN_NAME.SR4.

② The RUN_NAME of the transient simulation is used here.

③ The number of downwind positions where output is desired is entered here. For each downwind distance, questions 4 and 5 are required.

④ The x-coordinate is the downwind distance from the source where output is desired.

⑤ At each downwind distance, DEGADIS4 asks for the y- and z-coordinates desired. DEGADIS4 automatically supplies information about the centerline concentration ($y = 0$ and $z = 0$). Note that negative numbers signal that no more positions are desired for this particular downwind distance.

```
①   Enter the file name used for this run: BURRO9
②   Enter the number of downwind distances desired:
③    max of          10 downwind distances; 4 positions at each distance
    1

④   enter the x coordinate:
    100.
⑤     enter the y and z coordinate pairs at this distance:
    0.,1.
    30.,1.
    -1.,-1.
     tl:     18.000       tf:    99.263     dt:    2.0000     dist:     100.00
    FORTRAN STOP
    $
```

Notes on DEGADIS4 Output

① DEGADIS4 indicates the constants used in the x-direction dispersion correction. The user has the option of not making the x-direction dispersion correction if desired by making the appropriate changes to RUN_NAME.ER3.

② At each downwind distance specified, the concentration profile parameters are output as a function of time. Note that part of the output has been removed.

③ DEGADIS4 outputs the concentration time history at the off-centerline positions specified.

```
①   X-Direction correction was applied.
        Coefficient:          2.00000E-02
        Power:                 1.2200
        Minimum Distance:      130.00      m
②   Centerline values for the position -->
x:   100.00     m


  Time      Mole    Concentration Density      Gamma   Temperature   Half        Sz          Sy      Width at z=  1.00 m to:
           Fraction                                                  Width                           5.00     mole%  15.0     mole%
   (s)               (kg/m**3)  (kg/m**3)                  (K)        (m)        (m)         (m)        (m)           (m)
  20.0      0.264     0.192       1.1202     0.203        264.       13.4       0.984       17.2       28.6
  22.0      0.370     0.282       1.1350     0.185        248.       17.0       0.931       18.3       36.5

  24.0      0.455     0.367       1.1525     0.191        235.       18.9       0.899       18.7       40.7          27.2
  26.0      0.506     0.426       1.1652     0.194        227.       19.4       0.880       18.7       42.1          29.9
  28.0      0.529     0.451       1.1714     0.197        223.       19.6       0.868       18.6       42.5          30.7

  30.0      0.529     0.449       1.1678     0.191        224.       19.6       0.867       18.6       42.5          30.6
  32.0      0.529     0.450       1.1704     0.196        223.       19.6       0.868       18.6       42.5          30.6
  34.0      0.530     0.452       1.1706     0.195        223.       19.6       0.868       18.6       42.5          30.7
                                                      •

                                                      •

                                                      •

  90.0      0.530     0.452       1.1694     0.193        223.       19.6       0.869       18.6       42.5          30.7
  92.0      0.531     0.453       1.1702     0.194        223.       19.6       0.869       18.6       42.5          30.8
  94.0      0.529     0.448       1.1678     0.191        224.       19.6       0.869       18.7       42.5          30.7

  96.0      0.514     0.431       1.1637     0.189        226.       19.8       0.880       18.9       42.9          30.6
③        Time                 Mole fraction at:          Mole fraction at:           Mole fraction at:          Mole fraction at:
                              y=   0.00000E+00 m         y=     30.000     m         y=   -1.0000     m         y=   0.00000E+00 m
         (s)                  z=     1.0000    m         z=     1.0000     m         z=   -1.0000     m         z=   0.00000E+00 m
      18.00000                  0.0000000E+00              0.0000000E+00
      20.00000                  0.1086350                  4.4177441E-02
      22.00000                  0.1464779                  9.1616339E-02

      24.00000                  0.1790154                  0.1299436
      26.00000                  0.1982250                  0.1492197
      28.00000                  0.2065024                  0.1560519

      30.00000                  0.2052136                  0.1552692
      32.00000                  0.2060279                  0.1555801
      34.00000                  0.2068415                  0.1560749
                                                   •

                                                   •

                                                   •

      90.00000                  0.2070588                  0.1563020
      92.00000                  0.2075295                  0.1565796
      94.00000                  0.2058755                  0.1559159

      96.00000                  0.2017301                  0.1552051
$
```

(This page is intentionally left blank.)

APPENDIX C

JET-PLUME MODEL SOURCE CODE

(JETPLU_IN and JETPLU_MAIN)

(This page is intentionally left blank.)

```
        Program jetplu_in
c
c
c       JETPLU_IN is designed to perform two tasks including:
c
c       a) Read the input file for the JETPLU/DEGADIS model input.  From
c       this information, JETPLU_IN prepares the necessary file to run the
c       JETPLU/DEGADIS model.
c
c       b) Generate a command file to execute the JETPLU/DEGADIS model from
C       the input information.  This portion is highly dependent on the
c       VAX/VMS environment because of the manipulation of character strings
c       and the instructions necessary to invoke the VAX/VMS commands.
c
c.......................................................................
c
c       This program can be invoked interactively or in a command file
c       submitted to batch.  For either case, the syntax is:
c
c       $ RUN SYS$DEGADIS:JETPLU_IN
c       RUN_NAME
c
c       where RUN_NAME is the file name given to the particular simulation.
c       The input file RUN_NAME.IN is to be created before this program
c       is invoked using free formats.  The order of input is as follows:
c
c       <TITLE1>
C       <TITLE2>
C       <TITLE3>
C       <TITLE4>
C
C       <UO>  <ZO>
C       <ZR>
C       <INDVEL>  <ISTAB>  <RML>
C       <TAMB>  <PAMB>  <RELHUM>
C       <TSURF>
C
C       <GASNAM>
C       <GASMW>
C       <AVTIME>
C       <TEMJET>
C       <GASUL>  <GASLL>  <ZLL>
C       <INDHT>  <CPK>  <CPP>
C       <NDEN>
C
C       <ERATE>
C       <ELEJET>  <DIAJET>
C       <TEND>
c       <DISTMX>
C
C       Note that for readability, blank lines were inserted between the
```

```
c       input sections specifying a simulation title, atmospheric conditions,
c       gas properties, and the particular release conditions.  Symbol
c       definitions are as follows:
c
c       <TITLE1>, <TITLE2>, <TITLE3>, and <TITLE4> are four lines of up
c               to 80 characters each of a title for this simulation.
c
c       <U0> (m/s) is the ambient wind velocity at <Z0> (m).
c
c       <ZR> is the surface roughness (m).
c
c       <INDVEL> is an indicator which determines the method of calculation
c               for the ambient velocity profile in the jet/plume model as
c               follows:
c
c               For <INDVEL>=1, the Pasquill-Gifford stability category
c                  (in <ISTAB> using 1 for A, 2 for B, etc.) is used
c                  along with <ZR> to determine the Monin-Obukhov length
c                  <RML>; the log velocity profile is then determined
c                  using <RML>.
c
c               For <INDVEL>=2, the Monin-Obukhov length <RML> is supplied
c                  by the user; the log velocity profile is then determined
c                  using <RML>.  Note that <ISTAB> must still be specified.
c
c       <TAMB>, <PAMB>, and <RELHUM> are the ambient temperature (K), the
c               ambient pressure (atm or N/m**2), and the relative humidity (%),
c               respectively.
c
c       <TSURF> is the surface temperature (K); if <TSURF> < 250 K, <TSURF> is
c               set to <TAMB>.
c
c       <GASNAM> is a three-letter designation for the contaminant's name.
c               Any character string of three letters or less is valid; this
c               is for user run identification and does not access property
c               data.
c
c       <GASMW> is the contaminant's molecular weight (kg/kmole).
c
c       <AVTIME> is the averaging time (s).  This parameter is
c               used to estimate the value of <DELTAY>.
c
c       <TEMJET> is the temperature of the jet (K).
c
c       <GASUL> and <GASLL> are the concentrations to be used for estimating
c               contours for an upper and lower concentration level in
c               DEGADIS.  The calculations are made for the elevation
c               <ZLL>.  Note that the JETPLU/DEGADIS computations will be carried
c               out to <GASLL>/2.
c
c       <INDHT> is used to include heat transfer in the DEGADIS computations.
```

```
c                    Heat transfer is not included for <INDHT>=0.  For <INDHT>=1,
c                    heat transfer is included, and the heat transfer coefficient
c                    is calculated by DEGADIS.  <CPP> and <CPK> are used to
c                    calculate the heat capacity as a function of
c                    temperature according to the correlation included in
c                    DEGADIS.  If a constant heat capacity is
c                    desired, set <CPP> to 0. and <CPK> to the desired heat
c                    capacity (J/kg K).
c
c          <NDEN>   is used to specify the contaminant density profile.
c                   There are three cases for <NDEN>:
c
c                   <NDEN> = -1; The simulation treats the contaminant as if
c                          it were an ideal gas with a molal heat capacity
c                          equal to that of air.  Water condensation effects
c                          are ignored.  (equivalent to ISOFL=1 in DEGADIS)
c
c                   <NDEN> = 0;  The simulation treats the contaminant as if
c                          it were an ideal gas with the heat capacity indicated
c                          by <CPK> and <CPP>.  Water condensation effects
c                          are taken into account as appropriate.
c                          (equivalent to ISOFL=0 in DEGADIS)
c
c                   <NDEN> > 0; <NDEN> specifies the number of triples which follow
c                          in the next <NDEN> lines.  The triples are used to
c                          specify the contaminant concentration as a function of
c                          density based on adiabatic mixing with ambient air.
c                          The ordered triples represent (in order):
c                            (1) the contaminant mole fraction
c                            (2). the contaminant concentration (kg contam/m3 mix)
c                            (3) the mixture density (kg mixture/m3 mixture)
c                          The ordered triples must go from pure air to pure
c                          contaminant.  (equivalent to ISOFL=1 in DEGADIS)
c
c        <ERATE> is the mass evolution (release) rate (kg/s).
c
c        <ELEJET> is the inital jet elevation (m); the minimum jet elevation
c               is twice the surface roughness.  <DIAJET> is the initial
c               jet diameter (m).
c
c        <TEND> is the duration of the primary release (s).  For steady-state
c               releases, set <TEND> to 0.; to run the jet/plume model only, set
c               <TEND> to a negative number.
c
c        <DISTMX> is the maximum distance between output points in the JETPLU
c               output (m).
c
c-----------------------------------------------------------------------
c
c        T. Spicer
c        University of Arkansas
```

```
c      Department of Chemical Engineering
c      Fayetteville, AR 72701
c
c      (501) 575-4951
c
c------------------------------------------------------------------------
c

       Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

       include 'sys$degadis:DEGADISIN.dec'

       dimension DEN(5,igen)
C
       data vkc/0.35D0/
C
       logical check4
c
       character*80 TITLE(4)
C
       character*3 gas_name
C
       character*100 OPNRUP
       character OPNRUP1(100)
       equivalence (opnrup1(1),opnrup)
       character*4 IN,er1,er2,er3,com,scl,sr3,lis
       character*4 dummy
       character*3 plus, gasnam
       character*2 con
       DATA POUND/'//  '/,POUNDN/-1.E-20/
C
       DATA IN/'.IN '/,er1/'.er1'/,er2/'.er2'/,er3/'.er3'/
       data scl/'.scl'/,sr3/'.sr3'/,lis/'.lis'/
       data com/'.com'/
       data plus/' + '/,con/' -'/
c
C... GET THE FILE NAME TO BE USED BY ALL OF THE ROUTINES.........................
C
       READ(5,820) NCHAR,opnrup
       opnrup = opnrup(1:nchar) // in(1:4)
C
C... Now get the rest of the desired information from RUN_NAME.IN
C
       open(unit=8,name=opnrup,type='old')

       read(8,8000) title(1)
       read(8,8000) title(2)
       read(8,8000) title(3)
       read(8,8000) title(4)
 8000  format(a80)
 8010   format(a3)
        read(8,*) u0, z0
        read(8,*) zr

        read(8,*) indvel, istab, rml
c
c... Based on INDVEL, set the Monin-Obukhov length as desired.  Also,...........
c       calculate USTAR
c
        if( indvel.ne.1 .or. indvel.ne.2) indvel = 1
        if( istab.le.0 .or. istab.gt.6) istab = 4
        if(indvel .eq. 1) then
                if(istab.eq.1) then
                        rml = -11.43D0 * zr**0.103D0
                else if(istab.eq.2) then
                        rml = -25.98D0 * zr**0.171D0
                else if(istab.eq.3) then
                        rml = -123.4D0 * zr**0.304D0
                else if(istab.eq.4) then
                        rml = 0.0D0
                else if(istab.eq.5) then
                        rml = 123.4D0 * zr**0.304D0
                else if(istab.eq.6) then
                        rml = 25.98D0 * zr**0.171D0
                endif
        endif
        ustar = u0*vkc/(dlog((z0+zr)/zr) - psif(z0,rml))


        read(8,*) tamb, pamb, relhum

        if( tamb.le.0. ) tamb = tamb+273.15D0
        if( pamb.gt.1.1 ) pamb = pamb/101325.D0
        if( relhum.lt.0. .or. relhum.gt.100. ) relhum = 50.
c
c... Calculate the absolute humidity HUMID
c
        vaporp = 6.0298D-3* exp(5407.D0*(1.D0/273.15D0 - 1.D0/tamb))
        sat = 0.622D0*vaporp/(pamb - vaporp)
        humid = relhum/100.D0 * sat

        read(8,*) tsurf
        if( tsurf.lt.250. ) tsurf = tamb

        read(8,8010) gasnam
        read(8,*) gasmw
        read(8,*) avtime
        read(8,*) TEMJET
        read(8,*) gasul, gasll, zll
        if( gasll.le.0. ) gasll = 0.01
        if( gasul.le.gasll ) gasul = dmax1( 1.1D0*gasll, 1.0D0)
```

```
c
c... Now that AVTIME has been set, the value of DELTAY can be fixed.  Also
c       set the values of BETAY, DELTAZ, BETAZ, and GAMMAZ
c
        goto(161,162,163,164,165,166) istab
 161    timeav = dmax1( avtime, 18.4D0)                 ! A
        deltay = 0.423D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        deltaz = 107.66D0
        betaz  = -1.7172D0
        gammaz = 0.2770D0
        goto 170
 162    timeav = dmax1( avtime, 18.4D0)                 ! B
        deltay = 0.313D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        deltaz = 0.1355D0
        betaz  = 0.8752D0
        gammaz = 0.0136D0
        goto 170
 163    timeav = dmax1( avtime, 18.4D0)                 ! C
        deltay = 0.210D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        deltaz = 0.09623D0
        betaz  = 0.9477D0
        gammaz = -0.0020D0
        goto 170
 164    timeav = dmax1( avtime, 18.3D0)                 ! D
        deltay = 0.136D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        deltaz = 0.04134D0
        betaz  = 1.1737D0
        gammaz = -0.0316D0
        goto 170
 165    timeav = dmax1( avtime, 11.4D0)                 ! E
        deltay = 0.102D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        deltaz = 0.02275D0
        betaz  = 1.3010D0
        gammaz = -0.0450D0
        goto 170
 166    timeav = dmax1( avtime, 4.6D0)                  ! F
        deltay = 0.0674D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        deltaz = 0.01122D0
        betaz  = 1.4024D0
        gammaz = -0.0540D0
C
 170    continue

c
```

```
c... Recover INDHT, CPK, and CPP.  If CPP is set to 0, then CPK contains........
c       the (constant) heat capacity.
c
        read(8,*) indht, CPK, cpp
        if(cpp .eq. 0.D0) then
                cpp = 1.D0
                CPK = CPK*gasmw - 3.33D4
        endif

c
c... recover NDEN and set ISOFL and DEN accordingly..............................
c
        read(8,*) nden
        if(nden .lt. -1) nden=-1

        if(nden .eq. -1) then
                isofl = 1
                rhoe = pamb*101325.D0*gasmw/8314.D0/TEMJET
                rhoe = pamb*
                    (1.D0+humid)/(0.002833D0 + 0.004553D0*humid)/tamb
                den(1,1) = 0.D0
                den(2,1) = 0.D0
                den(3,1) = rhoe
                den(4,1) = 0.D0
                den(5,1) = tamb
                den(1,2) = 1.D0
                den(2,2) = rhoe
                den(3,2) = rhoe
                den(4,2) = 0.D0
                den(5,2) = tamb
        else if(nden .eq. 0) then
                isofl = 0
        else
                isofl = 1

                do iii = 1,nden
                read(8,*) den(1,iii), den(2,iii), den(3,iii)
                den(4,iii) = 0.D0
                den(5,iii) = tamb
                enddo
        endif

        nden0 = nden
        if(nden .eq. -1) nden0 = 2


        read(8,*) erate
        read(8,*) elejet, DIAJET
        if(elejet .lt. 2.D0*zr) then
            elejet = 2.D0*zr
            write(6,*) 'JETPLU_IN: ELEJET has been increased to: ',elejet
```

```
      endif
      read(8,*) tend
      check4 = .true.
      if(tend .gt. 0.) check4 = .false.

      read(8,*) distmx
c
c... It is time to prepare the input file for the JETPLU model..................
c
      opnrup = opnrup(1:nchar) // '.ino'
      open(unit=1,name=opnrup,type='new')

      write(1,603) title(1)
      write(1,603) title(2)
      write(1,603) title(3)
      write(1,603) title(4)
c
c... atmospheric parameters
c
      write(1,*) u0, z0
      write(1,*) zr
      write(1,*) istab, rml, ustar
      write(1,*) tamb, pamb, humid, relhum, tsurf
      write(1,*) avtime, deltay, betay
      write(1,*) deltaz, betaz, gammaz
c
c... contaminant parameters
c
      write(1,*) gasmw
      write(1,*) temjet
      write(1,*) gasul, gasll, zll
      write(1,*) CPK, cpp
      write(1,*) nden, nden0
         if(nden0 .gt. 0) then

             do iii = 1,nden0
             write(1,*) (den(jjj,iii),jjj=1,5)
             enddo
         endif
      write(1,*) isofl
c
c... spill parameters
c
      write(1,*) erate
      write(1,*) elejet, diajet
c
c... coding parameters
c
      alfa1 = 0.028D0
      alfa2 = 0.37D0
      write(1,*) alfa1, alfa2
      write(1,*) distmx


 603  format(a80)
      close(unit=1)

c
c... Now, prepare the command file............................................
c
C
C... FORMATS
C
 820  FORMAT(Q,A40)
c
c
1210  format(a4)
      opnrup = opnrup(1:nchar) // com(1:4)
c
      open(unit=8,name=opnrup,type='new',
     $ carriagecontrol='list',recordtype='variable')
c
c*** Lines to start the jet/plume model and invoke DEGBRIDGE
c
      opnrup = opnrup(1:nchar) // '.ino for001'
      write(8,1100) (opnrup1(i),i=1,nchar+11)
1100  format('$ assign ',51a1)
      opnrup = opnrup(1:nchar) // '.out for003'
      write(8,1100) (opnrup1(i),i=1,nchar+11)
      opnrup = opnrup(1:nchar) // '.ind for002'
      write(8,1100) (opnrup1(i),i=1,nchar+11)
      write(8,1160)
1160  format('$ run sys$degadis:jetplu')
      write(8,1170)
1170  format('$ deassign for001',/,'$ deassign for002',/,
     .    '$ deassign for003')

c
c... Bypass DEGBRIDGE if this is an "JETPLU only" run
c
      if(tend .lt. 0.) goto 3000
      write(8,1180)
1180  format('$ run sys$degadis:degbridge')
      write(8,1290) (opnrup1(i),i=1,nchar)
c
c
c
c
      opnrup = opnrup(1:nchar) // er1(1:4)
c
      write(8,1250) (opnrup1(i),i=1,nchar+4)
1250  format('$ copy/log SYS$DEGADIS:example.er1 ',40a1)
```

```
        IF(u0 .eq. 0.) then
                write(8,1280)
                write(8,1290) (opnrup1(i),i=1,nchar)
                goto 1340
                endif
        opnrup = opnrup(1:nchar) // er2(1:4)
c
        write(8,1260) (opnrup1(i),i=1,nchar+4)
 1260   format('$ copy/log SYS$DEGADIS:example.er2 ',40a1)
        opnrup = opnrup(1:nchar) // er3(1:4)
c
        if(.not.check4) then            ! transient
c
                write(8,1270) (opnrup1(i),i=1,nchar+4)
 1270           format('$ copy/log SYS$DEGADIS:example.er3 ',40a1)
c
                write(8,1280)
 1280           format('$ run SYS$DEGADIS:DEGADIS1')
                write(8,1290) (opnrup1(i),i=1,nchar)
 1290           format(40a1)
                write(8,1300)
 1300           format('$ run SYS$DEGADIS:DEGADIS2')
                write(8,1290) (opnrup1(i),i=1,nchar)
                write(8,1320)
 1320           format('$ run SYS$DEGADIS:DEGADIS3')
                write(8,1290) (opnrup1(i),i=1,nchar)
c
        else
                write(8,1280)
                write(8,1290) (opnrup1(i),i=1,nchar)
c
                write(8,1330)
 1330           format('$ run SYS$DEGADIS:SDEGADIS2')
                write(8,1290) (opnrup1(i),i=1,nchar)
c
        endif

        opnrup = opnrup(1:nchar) // '.out' //
     .          plus(1:3) //opnrup(1:nchar) // scl(1:4) //
     .          plus(1:3) // opnrup(1:nchar) // sr3(1:4) // con(1:2)
        write(8,1370) (opnrup1(i),i=1,3*nchar+20)
 1370   format('$ copy/log ',100a1)
        opnrup = opnrup(1:nchar) // lis(1:4)
        write(8,1390) (opnrup1(i),i=1,nchar+4)
 1390   format('  ',40a1)
c
 1340   close(unit=8)

 3000   continue
        write(6,2099)
 2099   format(/,' JETPLU_IN - beginning command file.')
```

```
        opnrup = '@' // opnrup(1:nchar) // ' '
        istat = lib$do_command(opnrup)
        write(6,2100)
  2100  format(/,' ?JETPLU_IN? command file failed to start.')
 c
        CALL EXIT
        END
####
```

```
      PROGRAM JETPLU_MAIN
C
c-----------------------------------------------------------------------
c
c     This program calculates the trajectory and dilution of a steady
c     gas jet released at right angles to the wind.  The zone before Gaussian
c     profiles can be assumed for the velocity and concentration profiles
c     is termed the zone of flow establishment; calculations for this zone
c     are done in the routine SETJET.  The zone after Gaussian profiles can
c     be assumed to apply is termed the zone of established flow.  The
c     ordinary differential equations describing this zone are included in
c     MODEL; these are integrated with the routine RKGST.  Interactions
c     between the jet/plume and the ground are accounted for using the method
c     of images.
c
c-----------------------------------------------------------------------
c
c     T. Spicer
c     University of Arkansas
c     Department of Chemical Engineering
c     Fayetteville, AR  72701
c
c     501-575-6516
C
C
      Implicit real*8(a-h,o-z), integer*4(i-n)
c
      include 'sys$degadis:DEGADIS1.dec'
c
      external model,modout

      character*80 title(4)
      character*24 tinp
      character*1  stabil(6)

      dimension prmt(6)
      dimension aux(8,6)

      DIMENSION YR(6),dYR(6)

      common
     ./GEN2/ DEN(5,igen)
     ./parm/ u0,z0,zr,rml,ustar,vkc,gg,rhoe,rhoa,deltay,betay,gammaf,
     .         cclow
     ./com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     .  gas_ufl,gas_lfl,gas_zsp,gas_name
     ./comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     .  humsrc
     ./PHYS/ UA, deltaz,betaz,gammaz,rho, temp,yc
     ./COEFF/ ALFA1,ALFA2, sc,cd,delta
     ./rks/ rk1,rk2,rk3,rk4,rk5,rk6, rate,totrte,xmo,zmo
     ./sys/ sysz,sy,sz,sya,sza
     ./lfl/ cufl, clfl
c
       character*3 gas_name

       data vkc/0.35D0/
       data gg/9.81D0/

       data sc/1.42D0/
       data cd/0.2D0/
       data delta/2.15D0/

       data stabil/'A','B','C','D','E','F'/

c
C
C... READ IN DATA FROM FOR001.DAT .....................................

       OPEN (UNIT=1,STATUS='OLD')
       OPEN (UNIT=3,STATUS='NEW')

       READ(1,600) (title(j),J=1,4)
       READ(1,*) u0,z0
       READ(1,*) ZR
       read(1,*) istab, rml, ustar
       read(1,*) tamb, pamb, humid, relhum, tsurf
       read(1,*) avtime, deltay, betay
       read(1,*) deltaz, betaz, gammaz

       read(1,*) gas_mw
       read(1,*) gas_temp
       read(1,*) gas_ufl, gas_lfl, gas_zsp
       read(1,*) gas_cpk, gas_cpp
       read(1,*) nden, nden0
       if(nden0 .gt. 0) then
               do iii = 1,nden0
               read(1,*) (den(jjj,iii),jjj=1,5)
               enddo

               den(1,nden0+1) = 2.D0
               rhoe = den(3,1)
               rhoe = den(3,nden0)
       else
               rhoe = Pamb*
     .            (1.D0+humid)/(0.002833D0 + 0.004553D0*humid)/TAmb
               rhoe = pamb*101325.D0*gas_mw/8314.D0/gas_temp
       endif
       gas_rhoe = rhoe

       read(1,*) isofl
```

```
        read(1,*) erate
        QINIT = erate/rhoe
        read(1,*) elejet, diajet

        READ(1,*) ALFA1, ALFA2
        read(1,*) distmx
c

        gamma = (rhoe - rhoa)/rhoe
        cclow = (gas_lfl/4.)*pamb*101325.D0*gas_mw/8314.D0/tamb
c
c... initialize DEN if necessary ......................................
c
        wc = 1.D0
        wa = 0.D0
        enth = cpc(gas_temp)*(gas_temp - tamb)
        call setden(wc, wa, enth)

        call adiabat(2,twc,twa,gas_lfl,ya,clfl,r,w,t,p)
        call adiabat(2,twc,twa,gas_ufl,ya,cufl,r,w,t,p)

c
C
C... PRINT VALUES OF PARAMETERS INITIAL CONDITIONS ......................

        OPEN(UNIT=3,STATUS='NEW')
        iii = lib$date_time(tinp)
        write(3,700) tinp
        write(3,710) (title(j),J=1,4)
        write(3,720)
        write(3,730) u0,z0,zr
        write(3,740) stabil(istab)
        write(3,750) rml,ustar,tamb,pamb,humid,relhum
        write(3,760) avtime, deltay, betay, deltaz, betaz, gammaz
        write(3,770)
        write(3,780) gas_mw,gas_temp,gas_ufl,gas_lfl,gas_cpk,gas_cpp
        write(3,790) nden
        if(nden .gt. 0) then
                write(3,800)
                do iii = 1,nden
                write(3,810) den(1,iii), den(2,iii), den(3,iii)
                enddo
        endif
        write(3,820) isofl
        write(3,830)
        write(3,840) erate,elejet,diajet
        write(3,850)
        write(3,860) alfa1,alfa2,distmx
        write(3,870) gas_zsp,(100.D0*gas_lfl),(100.D0*gas_ufl)
C
```

```
c
C... CALCULATE WIND SPEED AT THE JET DISCHARGE HEIGHT ..................
c

      UA=USTar/vkc*(dLOG((elejet+ZR)/ZR)-PSIF(elejet,rml))

C
C... CALL SETJET TO CALCULATE THE INITIAL CONDITIONS .....................
C       FOR JETPLU MODEL INTEGRATION.
c
      DO 17 I=1,6
      dyr(i) = 1.D0
   17 YR(I)=0.D0

      YR(1) = rhoe
      CALL SETJET(YR,UA,QINIT,DIAJET,elejet)
      rho  = gas_rhoe
      temp = gas_temp
C
C
C... CALCULATEd jet parameters:
c       Yr(1) ... CENTERLINE CONCENTRATION
c       Yr(2) ... sysz (product of local lateral and vertical dimension)
c       Yr(3) ... theta (ANGLE OF TRAJECTORY)
c       Yr(4) ... uc (VELOCITY DECREMENT)
c       Yr(5) ... x (downwind distance)
c       Yr(6) ... zj (elevation)
c
      prmt(1) = 0.D0                          ! lower limit
      prmt(2) = 100000.                       ! upper limit
      prmt(3) = max(Yr(5)/20.D0, 1.D-30)      ! initial step
      prmt(4) = 0.0001                        ! error criteria
      prmt(5) = distmx                        ! maximum step size
      prmt(6) = distmx/2.D0                   ! approximate output step size

      ndim = 6
      call rkgst(prmt,yr,dyr,ndim,ihlf,model,modout,aux)
c
      if(ihlf.ge.10) then
            write(3,*) '----ihlf.ge.10----'
            dist = 0.D0
            write(2,*) dist
            stop 'ihlf.ge.10'
      endif
c

C

 600  FORMAT(a80)
C
```

```
700     format(1x,1x,'********************',35x,'JETPLU/DEGADIS v2.1 ',35x,
     .          '********************',//,1x,a24,/)
710     format(1x,a80)
720     format(//,' Ambient Meteorological Conditions...')
730     format( /,' Ambient windspeed at reference height: ',1pg13.5,' m/s',
     .          /,'                      Reference height: ',1pg13.5,' m',
     .          /,'                     Surface roughness: ',1pg13.5,' m')
740     format(/,'               Pasquill stability class: ',12x,A1)
750     format( /,'                  Monin-Obukhov length: ',1pg13.5,' m',
     .          /,'                     Friction velocity: ',1pg13.5,' m/s',
     .          /,'                   Ambient temperature: ',1pg13.5,' K',
     .          /,'                      Ambient pressure: ',1pg13.5,' atm',
     .          /,'                      Ambient humidity: ',1pg13.5,
     .          /,'                     Relative humidity: ',1pg13.5,' %')
760     format( /,'              Specified averaging time: ',1pg13.5,' s',
     .          /,'                                DELTAy: ',1pg13.5,
     .          /,'                                 BETAy: ',1pg13.5,
     .          /,'                                DELTAz: ',1pg13.5,
     .          /,'                                 BETAz: ',1pg13.5,
     .          /,'                                GAMMAz: ',1pg13.5)
770     format(//,' Contaminant Properties...')
780     format( /,'          Contaminant molecular weight: ',1pg13.5,
     .          /,'                   Initial temperature: ',1pg13.5,
     .          /,'               Upper level of interest: ',1pg13.5,
     .          /,'               Lower level of interest: ',1pg13.5,
     .          /,'                Heat capacity constant: ',1pg13.5,
     .          /,'                   Heat capacity power: ',1pg13.5)
790     format(/,' NDEN flag: ',I3)
800     format(1x,1x,'Mole fraction',1x,1x,'Concentration',1x,4x,'Density',
     .          /,1x,15x,4x,'(kg/m3)',4x,4x,'(kg/m3)',4x)
810     format(1x,3(1x,1pg13.5))
820     format(/,' ISOFL flag: ',I3)
830     format(//,' Release Properties...')
840     format( /,'                          Release rate: ',1pg13.5,' kg/s',
     .          /,'                   Discharge elevation: ',1pg13.5,' m',
     .          /,'                    Discharge diameter: ',1pg13.5,' m')
850     format(//,' Model Parameters...')
860     format( /,'                                 ALFA1: ',1pg13.5,
     .          /,'                                 ALFA2: ',1pg13.5,
     .          /,'                                DISTMX: ',1pg13.5,' m',//)
870     format(1x,96x,'-------',1x,'At z= ',1pg11.3,' m',1x,'-------',/,
     .          1x,2x,'Downwind',2x,1x,'Elevation',2x,4x,'Mole',4x,
     .          1x,'Centerline',1x,2x,'Density',3x,'Temperature',1x,
     .          2x,'Sigma y',3x,2x,'Sigma z',3x,4x,'Mole',4x,7x,'Width to:',/,
     .          1x,2x,'Distance',2x,12x,2x,'Fraction',2x,'Concentration',
     .          47x,2x,'Fraction',1x,1pE8.2,'mol% ',1pE8.2,'mol%',/,
     .          1x,4x,'(m)',5x,4x,'(m)',5x,12x,2x,'(kg/m3)',3x,
     .          2x,'(kg/m3)',3x,4x,'(K)',5x,
     .          4x,'(m)',5x,4x,'(m)',5x,12x,4x,'(m)',5x,4x,'(m)'/)

C
```

```
        CALL EXIT
        END

c-----------------------------------------------------------------
c
c
        SUBROUTINE model(sss,yr,dyr,prmt)

        Implicit real*8(a-h,o-z), integer*4(i-n)
c
        include 'sys$degadis:DEGADIS1.dec'
        parameter (rt2 = 1.414213562D0)         ! sqrt(2.)

        external syfun
c


        common
     ./GEN2/ DEN(5,igen)
     ./parm/ u0,z0,zr,rml,ustar,vkc,gg,rhoe,rhoa,deltay,betay,gammaf,
     .          cclow
     ./com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     .  gas_ufl,gas_lfl,gas_zsp,gas_name
     ./comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     .  humsrc
     ./PHYS/ UA, deltaz,betaz,gammaz,rho, temp,yc
     ./COEFF/ ALFA1,ALFA2, sc,cd,delta
     ./rks/ rk1,rk2,rk3,rk4,rk5,rk6, rate,totrte,xmo,zmo
     ./sys/ sysz,sy,sz,sya,sza

c
        character*3 gas_name

        dimension yr(6),dyr(6),prmt(6)
        DIMENSION A(4,5)

C
C
c...integrated variables.............................................
c
        cc    = yr(1)
        sysz  = abs(yr(2))
        theta = yr(3)
        uc    = yr(4)
        dist  = yr(5)
        zj    = yr(6)

c
c... Calculate UENTR so that E1 is set to zero when UC<0
c
        uentr = max(uc,0.D0)
```

```
c
c...and some hybrid variables
c
      ST   = SIN(theta)
      ST2  = ST*ST
      CT   = COS(theta)
      CT2  = CT*CT
c
c... Estimate the density
c
      call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enth,temp)
      gamma = (rho-rhoa)/cc
c
c...calculate the ambient sigmas.........................................
c
      sya = deltay*dist**betay
      sza = deltaz*dist**betaz * exp(gammaz*log(dist)**2)
      dsya = 0.
      dsza = 0.
      if(dist .gt. 1.D0) then
         dsya = sya/dist* betay * ct
         dsza = sza/dist*(betaz + 2.D0*gammaz*log(dist)) * ct
      endif
c
c...calculate the "constants"/properties for the ellipse including:.........
c       - the values for SY and SZ
c       - the curved perimeter and area of the ellipse
c       - the ambient windspeed averaged over z in the ellipse
c       - RK1 through RK6
c
c... Values for SY and SZ
c
      sstart = sqrt(sysz)
      sinc   = sstart/20.D0
      sylow  = 1.D-10
      syhigh = max(sysz, 10.D0)
      call limit(syfun, sstart, sinc, syhigh, sylow)

      tol = 0.0001D0
      call zbrent(sy, syfun, sylow, syhigh, tol, ierr)
      if(ierr .ne. 0) stop 'ZBRENT failed.'
      sz = sysz/sy
c
c
c...the curved perimeter,  area of the ellipse
c

      call ellips(sy*delta, sz*delta, sz*delta, pe, area)

c
```

```
c...the ambient windspeed averaged over z in the ellipse (UA).
c       Calculate ALPHA from two velocity points calculated using the log
c       wind profile.  The integrated profile based on ALPHA gives UA.
c       Don't let ZJ be less than ZR.  The 0.01 offset for ZTOP gets
c       around a divide by zero when calculating ALPHA.
c
      zj = max(zj, zr)

      ztop = zj + delta*sz*ct + 0.01D0
      utop = ustar/vkc*(dlog((ztop+zr)/zr) - psif(ztop,rml))
      umid = ustar/vkc*(dlog((zj  +zr)/zr) - psif(zj  ,rml))
      alpha = dlog(utop/umid) / dlog(ztop/zj)
      alpha1 = 1.D0+alpha

      zbot = zj - delta*sz*ct
      zbot = max(zbot, zr)
      ua = umid/(alpha1*(ztop-zbot)*zj**alpha)
     .         *(ztop**alpha1 - zbot**alpha1)

      vj = uc+ua*ct
      vj = max(vj, 0.D0)

c
c...some additional hybrid variables
c
      UA2   = UA*UA
      UACT  = UA*CT
      UAST  = UA*ST
      uauc  = ua*uc
      uc2   = uc*uc
      ucct  = uc*ct
      ucst  = uc*st
      ccsy  = cc*sy
      ccsz  = cc*sz
      ccsysz= cc*sy*sz
C
C... Gas Component Mass Balance Equation .....................................
C
      qqq = rk1*uact + rk2*uc
      rate = ccsysz*qqq

      A(1,1) = qqq * sysz
      A(1,2) = qqq * cc
      A(1,3) = - rk1 * ccsysz * uast
      A(1,4) = rk2 * ccsysz
      A(1,5) = 0.D0
C
C... Overall Mass Balance Equation ...........................................
C
      qqq = rk3*uact + rk4*uc
      totrte = rhoa*qqq*sysz + gamma*rate
      A(2,1) = 0.D0
      A(2,2) = rhoa * qqq
      A(2,3) = - rhoa * rk3 * sysz * uast
      A(2,4) = rhoa * rk4 * sysz

      e3 = rk3*ua*(sza*dsya + sya*dsza)
      e2 = alfa2*uact*abs(st)*pe
      A(2,5) = rhoa*(ALFA1*uentr*pe + e2 + e3)
C
C... Z-direction Momentum Balance Equation ....................................
C
      drag = cd * pe * rhoa/2.D0*uast**2

      rrr = 2.D0*rk2*uauc*ct + rk1*ua2*ct2 + rk6*uc2
      qqq = 2.D0*rk4*uauc*ct + rk3*ua2*ct2 + rk5*uc2
      zmo = rhoa*st*qqq*sysz + gamma*st*rrr*ccsysz

      A(3,1) = gamma*sysz*rrr*st
      A(3,2) = rhoa*qqq*st + gamma*cc*rrr*st
      A(3,3) =   rhoa*sysz*(ct*qqq + st*(
     .               - 2.D0*rk4*uauc*st - 2.D0*rk3*ua2*ct*st))
     .       + gamma*ccsysz*(ct*rrr + st*(
     .               - 2.D0*rk2*uauc*st - 2.D0*rk1*ua2*ct*st))
      A(3,4) =     rhoa*sysz*st*(2.D0*rk4*uact + 2.D0*rk5*uc)
     .         + gamma*ccsysz*st*(2.D0*rk2*uact + 2.D0*rk6*uc)
      A(3,5) = -RK1*gg*gamma*ccsysz - sign(1.D0,theta)*drag*CT

C
C... X-Direction Momentum Balance Equation ....................................
C
      rrr = rk1*ua2*ct2*ct + 2.D0*rk2*uauc*ct2 + rk6*uc2*ct
      qqq = rk3*ua2*ct2*ct + 2.D0*rk4*uauc*ct2 + rk5*uc2*ct
      xmo = rhoa*qqq*sysz + gamma*rrr*ccsysz

      A(4,1) = gamma*sysz*rrr
      A(4,2) = rhoa*qqq + gamma*cc*rrr
      A(4,3) =   rhoa*sysz*(-3.D0*rk3*ua2*ct2*st
     .         - 4.D0*rk4*uauc*ct*st - rk5*uc2*st)
     .       + gamma*ccsysz*(-3.D0*rk1*ua2*ct2*st
     .         - 4.D0*rk2*uauc*ct*st - rk6*uc2*st)
      A(4,4) =     rhoa*sysz*(2.D0*rk4*ua*ct2 + 2.D0*rk5*ucct)
     .         + gamma*ccsysz*(2.D0*rk2*ua*ct2 + 2.D0*rk6*ucct)
      A(4,5) = UA*A(2,5) + drag*ABS(ST)

C
C... SIMUL USED FOR MATRIX INVERSION
C
      nnn = 4
      call SIMUL(nnn,A,dyr)
c
```

```
c... Force any stray derivatives to be physically realistic.
c
      if(dyr(1) .gt. 0.D0) dyr(1) = 0.D0
      if(dyr(2) .lt. 0.D0) dyr(2) = 0.D0
c
c...d(x) and d(z) are calculated directly from cos(theta) and sin(theta),
c     respectively.
c
      dyr(5) = ct
      dyr(6) = st
c
      RETURN
      END

c-----------------------------------------------------------------------

      subroutine modout(sss,yr,dyr,ihlf,ndim,prmt)
c
      Implicit real*8(a-h,o-z), integer*4(i-n)
c
      include 'sys$degadis:DEGADIS1.dec'
c
      common
     ./parm/ u0,z0,zr,rml,ustar,vkc,gg,rhoe,rhoa,deltay,betay,gammaf,
     .          cclow
     ./com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     .  gas_ufl,gas_lfl,gas_zsp,gas_name
     ./PHYS/ UA, deltaz,betaz,gammaz,rho, temp,yc
     ./COEFF/ ALFA1,ALFA2, sc,cd,delta
     ./rks/ rk1,rk2,rk3,rk4,rk5,rk6, rate,totrte,xmo,zmo
     ./sys/ sysz,sy,sz,sya,sza
     ./lfl/ cufl,clfl

      dimension yr(6),dyr(6),prmt(6)

      character*3 gas_name

      data ooo1/0.D0/, ooo2/0.D0/
      data iip/0/
c
c
      cc    = yr(1)
      sysz  = yr(2)
      theta = yr(3)
      uc    = yr(4)
      dist  = yr(5)
      zj    = yr(6)
c
c... Estimate CZSP and YZSP at y'=0 and z=GAS_ZSP...............................
c     Estimate as though z' and z represent the same distances.  Note that
c     no contribution from either exponential is included if the location
```

```
c       is outside the plume (outside DELTA*SZ which means ARG<2.31125).
c
        arg1 = 0.5D0*((zj - gas_zsp)/sz)**2
        arg2 = 0.5D0*((zj + gas_zsp)/sz)**2
        if(arg1 .lt. 2.31125D0) then
                arg1 = exp(-arg1)
        else
                arg1 = 0.D0
        endif
        if(arg2 .lt. 2.31125D0) then
                arg2 = exp(-arg2)
        else
                arg2 = 0.D0
        endif
        czsp = cc*(arg1 + arg2)

        call adiabat(0,wc,wa,yzsp,ya,czsp,rho,wm,enth,temp)

        if(czsp .gt. cufl) then
                wufl = sy*sqrt(2.D0*log(czsp/cufl))
                wlfl = sy*sqrt(2.D0*log(czsp/clfl))
        else if(czsp .gt. clfl) then
                wufl = 0.D0
                wlfl = sy*sqrt(2.D0*log(czsp/clfl))
        else
                wufl = 0.D0
                wlfl = 0.D0
        endif
                wufl = min(wufl, delta*sy)
                wlfl = min(wlfl, delta*sy)
C
C... IF THE JET REACHES THE GROUND, STOP THE COMPUTATION.......................
C
        IF (zj.LE.0.D0) then
                cc    = cc    - zj*(occ    - cc   )/(ozj - zj)
                sy    = sy    - zj*(osy    - sy   )/(ozj - zj)
                sz    = sz    - zj*(osz    - sz   )/(ozj - zj)
                theta = theta - zj*(otheta - theta)/(ozj - zj)
                uc    = uc    - zj*(ouc    - uc   )/(ozj - zj)
                dist  = dist  - zj*(odist  - dist )/(ozj - zj)
                ua    = ua    - zj*(oua    - ua   )/(ozj - zj)
                yzsp  = yzsp  - zj*(oyzsp  - yzsp )/(ozj - zj)
                wlfl  = wlfl  - zj*(owlfl  - wlfl )/(ozj - zj)
                wufl  = wufl  - zj*(owufl  - wufl )/(ozj - zj)

                zj = 0.D0
                cc = 2.D0*cc

                call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enth,temp)

                WRITE(2,*) dist,cc,delta*sy
```

```
           if(wufl.eq.0.D0 .and. wlfl.eq.0.D0) then
                WRITE(3,734) dist,zj,yc,cc,rho,temp,sy,sz,yzsp
           else if(wufl.eq.0.D0 .and. wlfl.ne.0.D0) then
                WRITE(3,734) dist,zj,yc,cc,rho,temp,sy,sz,yzsp,wlfl
           else
                WRITE(3,734) dist,zj,yc,cc,rho,temp,sy,sz,yzsp,wlfl,wufl
           endif
                WRITE(6,9999) dist,zj,cc,sy,sz,uc,ua,theta
                prmt(5) = 1.
                return
        endif
c
c... store the present values for interpolation when ZJ=0
c
        occ    = cc
        osy    = sy
        osz    = sz
        otheta = theta
        ouc    = uc
        odist  = dist
        ozj    = zj
        oua    = ua
        oyzsp  = yzsp
        owlfl  = wlfl
        owufl  = wufl
c
c... Calculate the image contribution .........................................
c
        cimage = cc*exp(-0.5D0*(2.D0*zj/sz)**2)
        cc = cc + cimage

        call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enth,temp)
c
C... IF THE JET cc drops below cclow, stop the calculation......................
C
        IF (cc.Lt.cclow) then
           if(wufl.eq.0.D0 .and. wlfl.eq.0.D0) then
                WRITE(3,734) dist,zj,yc,cc,rho,temp,sy,sz,yzsp
           else if(wufl.eq.0.D0 .and. wlfl.ne.0.D0) then
                WRITE(3,734) dist,zj,yc,cc,rho,temp,sy,sz,yzsp,wlfl
           else
                WRITE(3,734) dist,zj,yc,cc,rho,temp,sy,sz,yzsp,wlfl,wufl
           endif
                WRITE(6,9999) dist,zj,cc,sy,sz,uc,ua,theta
c
c...    set DIST to 0 to preclude DEGADIS from continuing
                dist = 0.D0
                WRITE(2,*) dist,cc,delta*sy
                prmt(5) = 1.
                return
        endif
```

```
c
c... printed output
c
	if(dist .ge. ooo1+ooo2) then
		ooo1 = dist
		ooo2 = prmt(6)

	  if(wufl.eq.0.D0 .and. wlfl.eq.0.D0) then
		WRITE(3,734) dist,zj,yc,cc,rho,temp,sy,sz,yzsp
	  else if(wufl.eq.0.D0 .and. wlfl.ne.0.D0) then
		WRITE(3,734) dist,zj,yc,cc,rho,temp,sy,sz,yzsp,wlfl
	  else
		WRITE(3,734) dist,zj,yc,cc,rho,temp,sy,sz,yzsp,wlfl,wufl
	  endif
		WRITE(6,9999) dist,zj,cc,sy,sz,uc,ua,theta

		iip = iip + 1
		if(iip .eq. 3) then
			write(3,735)
			write(6,735)
			iip = 0
		endif
	endif

c
 734	FORMAT(1X,10(1pg11.3,1x),1pg11.3)
 735	format(1x)
9999	FORMAT (1x,5(1pg9.3,1x),1pg10.3,1pg9.3,1pg10.3,3(1pg9.3,1x),
     .		1pg10.3,1x,1pg9.3)
	return
	end


c
C
C-----------------------------------------------------------------------

	SUBROUTINE SETJET(YR,ua,QINIT,DIAJET,elejet)
C
C	THIS SUBROUTINE TRANSFORMS THE EXIT 'TOPHAT' VELOCITY PROFILE OF
C	A CONTINUOUS JET TO THE SIMILARITY (GAUSSIAN) FORM REQUIRED FOR
C	INPUT TO THE JETPLU MODEL - USES WIND TUNNEL DATA CORRELATIONS BY
C	Y. KAMOTANI AND I. GREBER, NASA CONTRACTOR REPORT CR-2392, 3/74.
c
c	This has been simplified for vertical releases only.
C

	Implicit real*8(a-h,o-z), integer*4(i-n)
c
	include 'sys$degadis:DEGADIS1.dec'
	parameter (rt2 = 1.41421356D0)		! sqrt(2.)
```

```
c
        common
     ./parm/ u0,z0,zr,rml,ustar,vkc,gg,rhoe,rhoa,deltay,betay,gammaf,
     .          cclow
     ./COEFF/ ALFA1,ALFA2, sc,cd,delta
     ./rks/ rk1,rk2,rk3,rk4,rk5,rk6, rate,totrte,xmo,zmo
     ./sys/ sysz,sy,sz,sya,sza


        DIMENSION YR(6)
c
c... initial values for CC and UJ ..........................................
c
        cc = yr(1)
        uj = qinit/pi/diajet**2*4.D0
c
c... Calculate the length of the zone of flow development based on analysis
c       of Pratte and Baines work in the ASCE Journal of the Hydraulics
c       Division.
c
c......SOD represents (S/D) or length of development zone (S) nondimensionalized
c       using the jet diameter (D).
c
        sod = 7.7D0*(1.D0 - exp(-0.48D0*sqrt(rhoe*uj/rhoa/ua)))

c
c...Based on RJ ("J" in Kamotani and Grebber) and FROUDE (Froude number), ......
c       calculate "ae" and "be" (represented herein as AJ and BJ).

        rj = (rhoe/rhoa)*(Uj/ua)**2
        delrho = rhoe - rhoa
        froude = 1.688D0
        if(DELRHO .GT. 0.) froude = rhoa*ua**2/gg/delrho/diajet
        froude = min(froude, 1.688D0)

        if(rj .le. 0.036D0) then
                aj = 18.519D0*rj
                bj = 0.4D0
        else if(rj .gt. 0.036D0  .and. rj .le. 10.D0) then
                aj = EXP(0.2476D0 +0.3016D0*log(froude) +0.24386D0*LOG(rj))
                bj = 0.4D0
        else if(rj .gt. 10.D0  .and. rj .le. 50.D0) then
                aj = EXP(0.405465D0 +0.13138D0*LOG(rj) +0.054931D0*log(rj)**2)
                bj = exp(-0.744691D0 - 0.074525D0*log(rj))
        else if(rj .gt. 50.D0  .and. rj .le. 600.D0) then
                aj = EXP(-2.55104D0 +1.49202D0*log(rj) -0.097623D0*log(rj)**2)
                bj = exp(-0.446718D0 - 0.150694D0*log(rj))
        else
           write(6,*) 'J exceeded in SETJET; Aj and Bj are extrapolated.'
                aj = EXP(1.44099D0 + 0.243045D0*log(rj))
                bj = exp(-0.446718D0 - 0.150694D0*log(rj))
        endif

        bji = 1.D0/bj
c
c...Estimate (z/D) (as ZOD) from AJ, BJ, and SOD using a Newton-Raphson ........
c       procedure.
c
        zod = sod

        iii = 0
 100    continue
        iii = iii+1
        if(iii .gt. 100) then
                stop 'SETJET loop failed'
        endif

        xod = (zod/aj)**bji
        fff = xod**2 + zod**2 - sod**2
        fffp= xod**2*2.D0/bj/zod + 2.D0*zod
        zodn = zod - fff/fffp
        check = abs((zodn-zod)/zod)
        if(check .gt. 0.00001) then
                zod = zodn
                goto 100
        endif
c
c... Calculate DIST, THETA, ZJ, and UC from ZOD and XOD.  Note that XOD is .....
c       subject to underflows which is cured on a VAX by (XOD = XOD).
c
        xod = xod
        dist = xod*diajet
        if(xod .gt. 0.D0) then
            slope = bj*zod/xod
            theta = atan(slope)
        else if(xod .eq. 0.D0) then
            theta = pi/2.D0
        else
            stop 'XOD < 0. in SETJET.'
        endif
        zj = zod*diajet + elejet
        uc = uj - ua*cos(theta)
c
c...Determine SY and SZ to close the material balance ..........................
c
        erate = qinit*rhoe
c
c... Calculate the constants RK1 to RK6; PPP "corrects" the area of
c       the assumed rectangle to the ellipse.
c
```

```
      ppp = sqrt(pi)/2.D0

      qqq = erf(delta /rt2 *ppp)
      rk1 = pi * qqq * (2.D0*qqq)

      qqq = erf(delta /rt2*sqrt(1.D0+sc) *ppp)
      rk2 = pi/(1.D0+sc) * qqq * (2.D0*qqq)

      rk3 = pi*delta**2

      qqq = erf(delta /rt2*sqrt(sc) *ppp)
      rk4 = pi/sc * qqq * (2.D0*qqq)

      qqq = erf(delta *sqrt(sc) *ppp)
      rk5 = pi/(2.D0*sc) * qqq * (2.D0*qqq)

      qqq = erf(delta /rt2*sqrt(1.D0+2.D0*sc) *ppp)
      rk6 = pi/(1.D0+2.D0*sc) * qqq * (2.D0*qqq)

c
c... Calculate SYSZ.
c
      sysz = erate/cc/(rk1*ua*cos(theta) + rk2*uc)

      sy = sqrt(sysz)
      sz = sy

      yr(1) = cc
      yr(2) = sysz
      yr(3) = theta
      yr(4) = uc
      yr(5) = max(dist, 1.D-30)
      yr(6) = zj


      write(6,*) 'JETPLU/SETJET initial conditions...'
      write(6,*) ' '
      write(6,*) '      u0: ',u0    ,'     z0: ',z0
      write(6,*) '   erate: ',erate ,'     uj: ',uj
      write(6,*) '    rhoe: ',rhoe  ,'   rhoa: ',rhoa
      write(6,*) '      rj: ',rj    ,' froude: ',froude
      write(6,*) ' '
      write(6,*) '    dist: ',dist  ,'     zj: ',zj
      write(6,*) '  diajet: ',diajet,'     uc: ',uc
      write(6,*) '      sy: ',sy    ,'     sz: ',sz
      write(6,*) '   theta: ',theta ,'    sod: ',sod
      write(6,*) ' '
      write(6,*) '     rk1: ',rk1   ,'    rk2: ',rk2
      write(6,*) '     rk3: ',rk3   ,'    rk4: ',rk4
      write(6,*) '     rk5: ',rk5   ,'    rk6: ',rk6
      write(6,*) ' '
```

```
        write(6,*) ' DIST          ZJ          ',
     .         'CC          SY          SZ          UC          UA          THETA'
        write(6,*) ' '
        RETURN
        END


c-----------------------------------------------------------------
c
        function syfun(sytry)
c
        implicit real*8(a-h,o-z), integer*4(i-n)

        common
     ./rks/ rk1,rk2,rk3,rk4,rk5,rk6, rate,totrte,xmo,zmo
     ./sys/ sysz,sy,sz,sya,sza


        sztry = sysz/sytry
C
c       syfun = sya**2 - sza**2 - sytry**2 + sztry**2
c... to improve the accuracy of the calculation...
c
        syfun = (sya-sza)*(sya+sza) + (sztry-sytry)*(sztry+sytry)

        return
        end
####
```

APPENDIX D

DEGADIS MODEL SOURCE CODE

AFGEN.FOR D-3
AFGEN2.FOR D-4
ALPH.FOR D-5
CRFG.FOR D-8
DEGADIS1.DEC D-13
DEGADIS1.FOR D-14
DEGADIS2.DEC D-22
DEGADIS2.FOR D-23
DEGADIS3.DEC D-27
DEGADIS3.FOR D-28
DEGADIS4.DEC D-32
DEGADIS4.FOR D-33
DEGADISIN.DEC D-38
DEGADISIN.FOR D-39
DOSOUT.FOR D-45
ESTRT1.FOR D-50
ESTRT2.FOR D-54
ESTRT2SS.FOR D-56
ESTRT3.FOR D-58
GAMMA.FOR D-59
GETTIM.FOR D-61
GETTIMDOS.FOR D-63
HEAD.FOR D-64
INCGAMMA.FOR D-70
IO.FOR D-74
IOT.FOR D-76
LIMIT.FOR D-93
NOBL.FOR D-94
OB.FOR D-97
PSIF.FOR D-100
PSS.FOR D-101
PSSOUT.FOR D-104
PSSOUTSS.FOR D-107
RIPHIF.FOR D-110
RKGST.FOR D-114
SDEGADIS2.FOR D-122
SERIES.FOR D-130
SORTS.FOR D-131
SORTS1.FOR D-134
SRC1.FOR D-139
SRTOUT.FOR D-148
SSG.FOR D-152
SSGOUT.FOR D-155
SSGOUTSS.FOR D-158
SSOUT.FOR D-161
SSSUP.FOR D-163
STRT2.FOR D-172
STRT2SS.FOR D-175
STRT3.FOR D-178
SURFACE.FOR D-180
SZF.FOR D-182
TPROP.FOR D-185
TRANS1.FOR D-199
TRANS2.FOR D-202
TRANS2SS.FOR D-204
TRANS3.FOR D-206
TRAP.FOR D-207
TS.FOR D-216
TUPF.FOR D-217
UIT.FOR D-222
ZBRENT.FOR D-224

(This page is intentionally left blank.)

```
C.....................................................................
C
C       THIS FUNCTION LINEARLY INTERPOLATES FROM THE GIVEN
C            PAIR OF DATA POINTS.
C
        FUNCTION AFGEN(TAB,X,SPEC)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C

        include 'sys$degadis:DEGADIS1.dec'
        common/nend/poundn,pound
c
        character*4 pound
        character*(*) SPEC
        DIMENSION TAB(1)
C
        IF(X .GE. TAB(1)) GO TO 95
        WRITE(lunlog,1100) x,spec
        AFGEN = TAB(2)
        RETURN
C
 95     continue
        ix = 1
  100   Ix = ix + 2
C
        IY = IX + 1
        IF( TAB(IX).EQ.POUNDN .AND. TAB(IY).EQ.POUNDN ) GO TO 500
        IF(X .GE. TAB(IX)) GO TO 100
C
        IXP = IX-2
        IYP = IXP + 1
C
        SL = (TAB(IY) - TAB(IYP))/(TAB(IX) - TAB(IXP))
        AFGEN = SL*(X - TAB(IXP)) + TAB(IYP)
        RETURN
C
  500   CONTINUE
        IX = IX-2
        IY = IY-2
        IXP = IX-2
        IYP = IY-2
C
        SL = (TAB(IY) - TAB(IYP))/(TAB(IX) - TAB(IXP))
        AFGEN = SL*(X - TAB(IXP)) + TAB(IYP)
C
 1100   FORMAT(2X,'?AFGEN? UNDERFLOW; argument: ',1pg13.5,5X,A20)
        RETURN
        END
####
```

```
C..............................................................
C
C       THIS FUNCTION LINEARLY INTERPOLATES FROM THE GIVEN
C           PAIR OF DATA POINTS.
C
        FUNCTION AFGEN2(XTAB,TAB,X,SPEC)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        include 'sys$degadis:DEGADIS1.dec'
        common/nend/poundn,pound
c
        character*4 pound
        character*(*) SPEC
        DIMENSION XTAB(1),TAB(1)
C
        IF(X .GE. XTAB(1)) GO TO 95
        WRITE(lunlog,1100) x,spec
        AFGEN2 = TAB(2)
        RETURN
C
 95     continue
        ix = 1
  100   Ix = ix + 1
C
        IF( XTAB(IX).EQ.POUNDN ) GO TO 500
        IF(X .GE. XTAB(IX)) GO TO 100
C
        IXP = IX-1
C
        SL = (TAB(IX) - TAB(IXP))/(XTAB(IX) - XTAB(IXP))
        AFGEN2 = SL*(X - XTAB(IXP)) + TAB(IXP)
        RETURN
C
 500    CONTINUE
        IX = IX-1
        IXP = IX-1
C
        SL = (TAB(IX) - TAB(IXP))/(XTAB(IX) - XTAB(IXP))
        AFGEN2 = SL*(X - XTAB(IXP)) + TAB(IXP)
C
 1100   FORMAT(2X,'?AFGEN2? UNDERFLOW; argument: ',1pg13.5,5X,A20)
        RETURN
        END
####
```

```
C.....................................................................
C
C       SUBROUTINES TO CALCULATE THE VALUE OF ALPHA
C
        SUBROUTINE ALPH

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        include 'sys$degadis:DEGADIS1.dec'
C
        COMMON
     $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/ALP/ALPHA,alpha1
     $/alphcom/ ialpfl,alpco
C
        REAL*8 ML,K
C
        EXTERNAL ALPHI
C
        PSI = PSIF(Z0,ML)
        USTAR = U0*K/(dLOG((Z0+ZR)/ZR) - PSI)
c
        if(u0 .eq. 0.) then
                alpha = 0.
                ustar = 0.
                return
                endif
c
        if(ialpfl.eq. 0) then
                alpha = alpco
                return
                endif
C
C*** ZBRENT USED TO DETERMINE THE ROOT OF THE REQUIRED INTEGRAL EQUATION
C
        IER = 0
C
        CALL zbrent(alpha, ALPHI, XLI, XRI, EPS, IER)
        IF( IER .NE. 0 ) CALL trap(19,IER)
C
        RETURN
        END
c
c
C.....................................................................
C
C       FUNCTION TO EVALUATE THE WEIGHTED EUCLIDEAN NORM OF THE
```

```
C        ERROR ASSOCIATED WITH THE POWER LAW FIT OF THE WIND PROFILE.
C
        FUNCTION ALPHI(X)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        include 'sys$degadis:DEGADIS1.dec'
C
        COMMON
     $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/ALP/ALPHA,alpha1
C
        REAL*8 ML,K
C
        DIMENSION Y(1),DERY(1),PRMT(5),AUX(8)
        EXTERNAL ARG,ARGOUT
C
        ALPHA = X
C
        PRMT(1) = ZO
        PRMT(2) = dmax1(ZLOW,zr)        ! to take care of large zr
        PRMT(3) = STPINZ
        PRMT(4) = ERBNDZ
        PRMT(5) = stpmxz
C
        Y(1) = 0.0D00
C
        DERY(1) = 1.0D00
C
        NDIM = 1
        IHLF = 0
C
        CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,ARG,ARGOUT,AUX)
C
        IF(IHLF .GE. 10) CALL trap(18,IHLF)
        ALPHI = Y(1)
        RETURN
        END
c
c
C.................................................................
C
C       FUNCTION TO EVALUATE THE ARGUMENT OF THE INTEGRAL EXPRESSION
C
        SUBROUTINE ARG(Z,Y,D,PRMT)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
```

```
C
      include 'sys$degadis:DEGADIS1.dec'
C
      COMMON
     $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/ALP/ALPHA,alpha1
     $/alphcom/ ialpfl,alpco
C
      REAL*8 ML,K
C
      DIMENSION Y(1),D(1),PRMT(1)
C
C*** WEIGHT FUNCTION USED
C
      W = 1.D00/(1.D00 + Z)
      if(ialpfl.eq. 2) w= 1.D00
C
C*** WIND VELOCITY @ Z -- BEST FIT
C
      UBST = USTAR/K*(dLOG((Z+ZR)/ZR) - PSIF(Z,ML))
C
C*** WIND VELOCITY @ Z -- POWER LAW APPROXIMATION
C
      UALP = U0 * (Z/Z0) ** ALPHA
C
      D(1) = W * (UBST - UALP) * dLOG(Z/Z0) * UALP
      RETURN
      END
c
C
      SUBROUTINE ARGOUT
      RETURN
      END
####
```

```
C..............................................................
C
C       SUBROUTINE TO CREATE RADG,QSTR,srcden,srcwc,srcwa,srcenth DATA LISTS
C
C       PARAMETERS  --  TABLE - WORKSPACE VECTOR
C                       NTAB - DIMENSION OF TABLE DIVIDED BY iout_src
C                       RER - RELATIVE ERROR BOUND OF CREATED
C                               DATA PAIRS BY LINEAR INTERPOLATION
C
C       VALUES OF TIME, RADG, height, QSTR, SZ0, yc, ya, rho, Ri,
c               wc,wa,enthalpy,temp
c         ARE READ INTO
C         TABLE(1) TO TABLE(13) RESPECTIVELY.
C
C..............................................................
C
        SUBROUTINE CRFG(TABLE,NTAB,rer)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        DIMENSION TABLE(1)
C
        include 'sys$degadis:DEGADIS1.dec'
        parameter (zero= 1.e-20)
c
        COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/NEND/ POUNDN,POUND
c
        character*4 pound
c
        LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
        DATA NI/1/
c
        data iti/1/     ! time - element no 1 in record
        data irg/2/     ! Radg - element no 2 in record
        data iqs/4/     ! Qstar - element no 4 in record
        data idn/8/     ! rho - element no 8 in record
        data iwc/10/    ! wc - element no 10 in record
        data iwa/11/    ! wa - element no 11 in record
        data ien/12/    ! enthalpy - element no 12 in record
c
C
C       OUTPUT CREATED VECTORS TO A PRINT FILE
```

```
C
C
      READ(9,*) (TABLE(J),J=1,iout_src)
C
      WRITE(8,1111)
      WRITE(8,1105)
      if(isofl.eq. 1) then
      WRITE(8,1102)
      WRITE(8,1103)
      WRITE(8,1140) (TABLE(J),J=1,6),table(8),table(9)
      else
      WRITE(8,1100)
      WRITE(8,1104)
      WRITE(8,1140) (TABLE(J),J=1,6),table(8),table(13),table(9)
      endif
      ispace = 1
C
 1100 FORMAT(/,5X,'Time',5X,2x,'Gas Radius',2x,4X,'Height',4X,
     $4x,'Qstar',5x,2x,'SZ(x=L/2.)',2x,1X,'Mole frac C',2x,
     $3x,'Density',4x,1x,'Temperature',2x,3x,'Rich No.',3x)
 1102 FORMAT(/,5X,'Time',5X,2x,'Gas Radius',2x,4X,'Height',4X,
     $4x,'Qstar',5x,2x,'SZ(x=L/2.)',2x,1X,'Mole frac C',2x,
     $3x,'Density',4x,3x,'Rich No.',3x)
 1103 FORMAT(1H ,4X,'sec',6X,6X,'m',7X,6X,'m',7X,
     $2X,'kg/m**2/s',3X,6X,'m',7X,14x,3x,'kg/m**3',4x,14x,/)
 1104 FORMAT(1H ,4X,'sec',6X,6X,'m',7X,6X,'m',7X,
     $2X,'kg/m**2/s',3X,6X,'m',7X,14x,3x,'kg/m**3',4x,6x,'K',7x,14x,/)
 1105 FORMAT(1H ,23X,'*****',21X,'CALCULATED SOURCE PARAMETERS',21X,
     $'*****')
C
      RADG(1,1) = 0.
      RADG(2,1) = TABLE(2)
      QSTR(1,1) = 0.
      QSTR(2,1) = TABLE(4)
      srcden(1,1) = 0.
      srcden(2,1) = table(8)
      srcwc(1,1) = 0.
      srcwc(2,1) = table(10)
      srcwa(1,1) = 0.
      srcwa(2,1) = table(11)
      srcenth(1,1) = 0.
      srcenth(2,1) = table(12)
C
      READ(9,*) (TABLE(J),J=1,iout_src)
      L = 2
C
C*** L IS THE NUMBER OF RECORDS WHICH HAVE BEEN READ
C
  100 CONTINUE
      DO 120 I=2,NTAB
C
```

```
C*** MOVE LAST RECORD READ INTO THE LAST ACTIVE POSITION OF TABLE
C
      DO 130 J = 1,iout_src
      KK = iout_src * (I-1) + J
  130 TABLE(KK) = TABLE(J)
      KK = iout_src * I
C
C*** READ THE NEXT RECORD.  INCREMENT L.
C
      L = L + 1
      READ(9,*,END=900) (TABLE(J),J=1,iout_src)
C
C
      DO 140 Kkk = 2,I
C
      KT    = iout_src*(Kkk-1) + iti
      KRG   = iout_src*(Kkk-1) + irg
      KQSTR = iout_src*(Kkk-1) + iqs
      KCA   = iout_src*(Kkk-1) + idn
      KwC   = iout_src*(Kkk-1) + iwc
      Kwa   = iout_src*(Kkk-1) + iwa
      Ken   = iout_src*(Kkk-1) + ien
C
      timeslot = radg(1,ni)
      ratio = (table(kt) - timeslot) / (table(iti) - timeslot)
c
      ANSRG = (TABLE(irg) - RADG(2,NI)) * ratio + RADG(2,NI)
      ANSQ = (TABLE(iqs) - QSTR(2,NI)) * ratio + QSTR(2,NI)
      ANSCA = (TABLE(idn) - srcden(2,NI)) * ratio + srcden(2,NI)
      ANSwC = (TABLE(iwc) - srcwc(2,NI)) * ratio + srcwc(2,NI)
      ANSwa = (TABLE(iwa) - srcwa(2,NI)) * ratio + srcwa(2,NI)
      ANSen = (TABLE(ien) - srcenth(2,NI)) * ratio + srcenth(2,NI)
C
      ERRG = ABS(ANSRG - TABLE(KRG))/TABLE(KRG)
      ERQSTR = ABS(ANSQ - TABLE(KQSTR))/(TABLE(KQSTR)+zero)
      ER0 = dMAX1(ERRG,ERQSTR)
c
      ERCA = ABS(ANSCA - TABLE(KCA))/TABLE(KCA)
      ER0 = dMAX1(ER0,ERCA)
c
      ERwC = ABS(ANSwC - TABLE(KwC))/(TABLE(KwC)+ zero)
      ER0 = dMAX1(ER0,ERwC)
      ERwa = ABS(ANSwa - TABLE(Kwa))/(TABLE(Kwa)+ zero)
      ER0 = dMAX1(ER0,ERwa)
      ERen = ABS(ANSen - TABLE(Ken))/(TABLE(Ken)+ zero)
      ER0 = dMAX1(ER0,ERen)
C
      IF(ER0 .GT. RER) GO TO 150
  140 CONTINUE
  120 CONTINUE
C
```

```
        WRITE(lunlog,1110)
C
C*** RECORD NEXT DATA PAIR. SINCE ERROR EXCEEDED, RECORD THE LAST
C***  DATA PAIR WHICH SATISFIED THE ERROR CRITERIA WHICH IS STORED
C***  IN TABLE(KK-(iout_src-1)) TO TABLE(KK)
C
  150   NI = NI + 1
c
        KT    = KK - iout_src + iti
        KRG   = KK - iout_src + irg
        KQSTR = KK - iout_src + iqs
        KCA   = KK - iout_src + idn
        KwC   = KK - iout_src + iwc
        Kwa   = KK - iout_src + iwa
        Ken   = KK - iout_src + ien
c
        IF(NI .GT. MAXL) then
                write(lunlog,*) ' CRFG? Time out: ',table(kt)
                CALL trap(5)
                endif
C
        RADG(1,NI)   = TABLE(KT)
        RADG(2,NI)   = TABLE(KRG)
        QSTR(1,NI)   = TABLE(KT)
        QSTR(2,NI)   = TABLE(KQSTR)
        srcden(1,NI) = TABLE(KT)
        srcden(2,NI) = TABLE(KCA)
        srcwc(1,NI)  = TABLE(KT)
        srcwc(2,NI)  = TABLE(KWC)
        srcwa(1,NI)  = TABLE(KT)
        srcwa(2,NI)  = TABLE(KWA)
        srcenth(1,NI)  = TABLE(KT)
        srcenth(2,NI)  = TABLE(KEN)
C
C*** WRITE THE POINTS JUST RECORDED TO UNIT=8
C
        if(isofl.eq. 1) then
        WRITE(8,1140) (TABLE(J),J=kt,kt+5),table(kt+7),table(kt+8)
        else
        WRITE(8,1140) (TABLE(J),J=kt,kt+5),table(kt+7),
        1       table(kt+12),table(kt+8)
        endif
        ispace = ispace + 1
        if(ispace.eq. 3) then
                ispace = 0
                write(8,1111)
                endif
C
        GO TO 100
C
  900   CONTINUE                        ! EOF encountered
```

```
C
      NI   = NI   + 1
      IF(NI+1 .GT. MAXL) then
            write(lunlog,*) ' CRFG? Time out: ',table(iti)
            CALL trap(5)
            endif
c
      RADG(1,NI)   = TABLE(iti)
      RADG(2,NI)   = TABLE(irg)
      QSTR(1,NI)   = TABLE(iti)
      QSTR(2,NI)   = TABLE(iqs)
      srcden(1,NI) = TABLE(iti)
      srcden(2,NI) = TABLE(idn)
      srcwc(1,NI)  = TABLE(iti)
      srcwc(2,NI)  = TABLE(iwc)
      srcwa(1,NI)  = TABLE(iti)
      srcwa(2,NI)  = TABLE(iwa)
      srcenth(1,NI)  = TABLE(iti)
      srcenth(2,NI)  = TABLE(ien)
c
      if(isofl.eq. 1) then
      WRITE(8,1140) (TABLE(J),J=1,6),table(8),table(9)
      else
      WRITE(8,1140) (TABLE(J),J=1,6),table(8),table(13),table(9)
      endif
      ispace = ispace + 1
      if(ispace.eq. 3) then
            ispace = 0
            write(8,1111)
            endif
c
      NI   = NI   + 1
      DO 910 I =1,2
      RADG(I,NI)    = POUNDN
      QSTR(I,NI)    = POUNDN
      srcden(I,NI)  = POUNDN
      srcwc(I,NI)   = POUNDN
      srcwa(I,NI)   = POUNDN
      srcenth(I,NI) = POUNDN
  910 CONTINUE
C
      RETURN
C
 1110 FORMAT(' ?CRFG? TABLE exceeded without point selection ',
     $'- execution continuing')
 1111 FORMAT(1H )
c1140 FORMAT(1H ,<iout_src>(1PG13.6,1X))
 1140 FORMAT(1H ,9(1PG13.6,1X))
      END
####
```

```
C.....................................................
C
C
C       DIMENSIONS/DECLARATIONS for DEGADIS1
C
        include 'sys$degadis:DEGADISIN.dec'
c
c  maxl is the length of the /GEN3/ output vectors
c
        parameter (     lunlog= 6,
     $                  sqrtpi= 1.77 245 3851D0,            ! sqrt(pi)
     $                  maxl= 800,
     $                  maxl2= 2*maxl,
     $                  iout_src= 13)
C
####
```

```
      PROGRAM DEGADIS1
C
C***********************************************************************
C***********************************************************************
C***********************************************************************
C
C       Program description:
C
C       DEGADIS1 estimates the ambient wind profile power alpha and
C       characterizes the primary gas source.
C
C
C       Program usage:
C
C       Consult Volume III of the Final Report to U. S. Coast Guard
C       contract DT-CG-23-80-C-20029 entitled "Development of an
C       Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures".
C
C       J. A. Havens
C       T. O. Spicer
C
C       University of Arkansas
C       Department of Chemical Engineering
C       Fayetteville, AR 72701
C
C       April 1985
C
C
C       This project was sponsored by the U. S. Coast Guard and the Gas
C       Research Institute under contract DT-CG-23-80-C-20029.
C
C
C       Disclaimer:
C
C       This computer code material was prepared by the University of
C       Arkansas as an account of work sponsored by the U. S. Coast Guard
C       and the Gas Research Institute.  Neither the University of Arkansas,
C       nor any person acting on its behalf:
C
C       a.  Makes any warranty or representation, express or implied,
C           with respect to the accuracy, completeness, or usefulness
C           of the information contained in this computer code material,
C           or that the use of any apparatus, method, numerical model,
C           or process disclosed in this computer code material may not
C           infringe privately owned rights; or
C
C       b.  Assumes any liability with respect to the use of, or for
C           damages resulting from the use of, any information,
C           apparatus, method, or process disclosed in this computer
C           code material.
C
```

```
C
C***********************************************************************
C***********************************************************************
C***********************************************************************
C
C
C       DIMENSIONS/DECLARATIONS
c

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADIS1.dec'
c
c  ntab is the dimension of table divided by iout_src
c
        parameter (     ntab0=910,
     $                  ntab=ntab0/iout_src)
c
        include '($ssdef)'
C
C       BLOCK COMMON
C
        COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/TITL/ TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/SZFC/ szstp0,szerr,szstpmx,szsz0
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl,swcl,swal,senl,srhl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/vucom/ vua,vub,vuc,vud,vudelta,vuflag
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/com_ENTHAL/ H_masrte,H_airrte,H_watrte
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/alphcom/ ialpfl,alpco
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     $/COM_SURF/ HTCUTS
     ./oomsin/ oodist,avtime
C
C
      character*80 TITLE(4)
C
      character*4 pound
      character*24 TSRC,TINP,TOBS,TSRT
      character*3 gas_name
C
      real*4 tt1
      REAL*8 ML,K
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      logical vuflag
      logical reflag
C
      REAL*8 L,LO
      DIMENSION PRNT(25),Y(7),DERY(7),AUX(8,7)
      EXTERNAL SRC1,SRC1O
      character*40 opnrup1
      character OPNRUP(40)
      character*4 INP,ER1,SCD,TR2,scl
c
      dimension table(ntab0)
c
      equivalence(opnrup(1),opnrup1)
c
C.........................................................
C
C     DATA
C
      DATA POUNDN/-1.D-20/,POUND/'//  '/
      DATA USTAR/0.D0/,GAMMAF/0.D0/
      DATA G/9.81D0/, K/0.35D0/
      DATA GAMMAF/0.D0/
      DATA PRNT/25*0.D0/
      DATA Y/7*0.D0/, DERY/7*0.D0/
      DATA TIME0/0.D0/,NDIM/0/
      DATA EMAX/0.D0/,TSC1/0.D0/
      DATA PTIME/igen*0.D0/
      DATA ET/igen*0.D0/,R1T/igen*0.D0/
      DATA PWC/igen*0.D0/,PTEMP/igen*0.D0/
      DATA PFRACV/igen*0.D0/,PENTH/igen*0.D0/
      DATA PRHO/igen*0.D0/
      data DEN/igen*0.D0,igen*0.D0,igen*0.D0,igen*0.D0,igen*0.D0/
c
      data reflag/.true./
C
      DATA INP/'.inp'/,ER1/'.er1'/
      DATA SCD/'.scd'/,TR2/'.tr2'/
      data scl/'.scl'/
C
```

```
C.....................................................
C
C       MAIN
C
C*** GET THE EXECUTION TIME
C
        t1 = secnds(0.)
        istat = lib$date_TIME(TSRC)
        if(istat .ne. ss$_normal) stop'lib$date_time failure'
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
c       WRITE(lunlog,1130)
c1130   FORMAT(5X,'Enter the file name being used for this run: ',$)
        read(5,1000) nchar,opnrup       ! unit 5 gets command file too
 1000   format(q,40a1)
c
        opnrup1 = opnrup1(1:nchar) // er1(1:4)
c
        CALL ESTRT1(OPNRUP1)
        HTCUTS = HTCUT
C
        opnrup1 = opnrup1(1:nchar) // inp(1:4)
        CALL IO(tend,gmass0,OPNRUP1)
        if(check4) srcoer = 2.8*srcoer
        CALL ALPH
C
        alpha1 = alpha+1.
        WRITE(lunlog,1105) ALPHA
 1105   FORMAT(5X,'THE VALUE OF ALPHA IS ',F6.4)
C
C
        GAMMAF = GAMMA(1./ALPHA1)
        TSC1 = TEND
C
c
c*** set the density and enthalpy functions in TPROP
c
        call setenthal(h_masrte,h_airrte,h_watrte)
        call setden(1.D00,0.D00,h_masrte)
C
C.....................................................
C
C       SOURCE INTEGRATION (CA = RHOE)
C
        opnrup1 = opnrup1(1:nchar) // scd(1:4)
c
        OPEN(UNIT=9,NAME=OPNRUP1,recl=202,TYPE='SCRATCH',
     $  carriagecontrol='list',
     $  recordtype='variable')
c
```

```
          gmass  = gmass0
    C
    C.....................................................
    C
    C       START THE GAS BLANKET?
    C
          L = 2.*AFGEN2(PTIME,R1T,TIME0,'R1T-MN')
          QSTRE = AFGEN2(PTIME,ET,TIME0,'ET-MN')/(pi*L**2/4.)
          PWCP = AFGEN2(PTIME,PWC,TIME0,'PWC')
          HPRIM = AFGEN2(PTIME,PENTH,TIME0,'PENTH')
          RHOP = AFGEN2(PTIME,PRHO,TIME0,'PRHO')
          CCP = PWCP*RHOP
    c
    c
          qstar = 0.
          if(u0 .ne. 0.)
         $  qstar = CCP*k*ustar*alpha1*dellay/(dellay-1.)/phihat(RHOP,L)
    c
    c
          write(lunlog,3010) time0,l,sz,qstar,qstre
          IF(QSTRE.lt.QSTaR .and. gmass0.eq.0.) then
                tsc1 = 0.
                GOTO 105
                endif
    C
      100 CONTINUE
          check3 = .false.
    C
    C*** INITIAL CONDITIONS
    C
          if(time0.ne. 0.) then
                L0 = 2.*AFGEN2(PTIME,R1T,TIME0,'R1T-MN')
                QSTRE = AFGEN2(PTIME,ET,TIME0,'ET-MN')/(pi*L0**2/4.)
          PWCP = AFGEN2(PTIME,PWC,TIME0,'PWC')
          HPRIM = AFGEN2(PTIME,PENTH,TIME0,'PENTH')
          RHOP = AFGEN2(PTIME,PRHO,TIME0,'PRHO')
          CCP = PWCP*RHOP
    c
          qstar =0.
          if(u0.ne. 0.)
         $  qstar = CCP*k*ustar*alpha1*dellay/(dellay-1.)/phihat(RHOP,L0)
    c
                endif
    C
    C*** SET UP INTEGRATION PARAMETERS
    C
    C*** VARIABLE        SUBSCRIPT
    C*** --------        ---------
    C***    RG             Y(1)
    C***    mass           Y(2)
    C***    massc          Y(3)
```

```
C***    massa           Y(4)
C***    Enthalpy        Y(5)
c***    moe             y(6)
C***    TIME            X
C
        PRMT(1) = TIMEO
        PRMT(2) = 6.023E23
        PRMT(3) = STPIN
        PRMT(4) = ERBND
        PRMT(5) = 4.DO*STPMX
C       PRMT(6) = EMAX -- OUTPUT
c
        do iii = 6,25
        prmt(iii) = 0.
        enddo
C
        Y(1) = AFGEN2(PTIME,R1T,TIMEO,'R1T-MN')
        rmax = y(1)
        Y(2) = dmax1( gmass, (pi*Y(1)**2*1.1*srccut*RHOP))
        vuflag = .false.
        htod = gmass/RHOP/2./pi/Y(1)**3
        prmt(22) = htod * 2.*Y(1)         ! initial height of the tail
        prmt(24) = prmt(22)
        prmt(23) = 0.                     ! initial height of the head
        prmt(25) = prmt(23)
        if(htod .gt. 0.1) vuflag = .true.
        Y(3) = Y(2) * PWCP
        PWAP = (1. - PWCP)/(1. + HUMID)           ! water from humidity only
        Y(4) = Y(2) * PWAP
        Y(5) = Y(2) * HPRIM
        y(6) = 0.0
C
        DERY(1) = WTRG
        DERY(2) = WTtm
        DERY(3) = WTyc
        DERY(4) = WTya
        DERY(5) = WTeb
        dery(6) = wtmb
C
        NDIM = 6
C
C*** PERFORM INTEGRATION
C
        WRITE(lunlog,1145)
 1145   FORMAT(5X,'Beginning Integration Step - Gas blanket')
C
        CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,SRC1,SRC1O,AUX)
C
        IF(IHLF .GT. 10) CALL trap(1,IHLF)
        IF(CHECK3) then
              TEND = TSC1
```

```
                GO TO 110
         end if
C
C..........................................................
C
C        RESTART THE GAS BLANKET?
C
         TIME0 = TSC1
         L = 2.*AFGEN2(PTIME,R1T,TIME0,'R1T-MN')
         QSTRE = AFGEN2(PTIME,ET,TIME0,'ET-MN')/(pi*L**2/4.)
         PWCP = AFGEN2(PTIME,PWC,TIME0,'PWC')
         RHOP = AFGEN2(PTIME,PRHO,TIME0,'PRHO')
         CCP = PWCP*RHOP
c
         qstar = 0.
         if(u0.ne. 0.)
     $    qstar = CCP*k*ustar*alpha1*dellay/(dellay-1.)/phihat(RHOP,L)
c
         write(lunlog,3010) time0,l,sz,qstar,qstre
 3010    format(//,' time0: ',1pg13.5,t40,'l: ',1pg13.5,
     $/,' sz0: ',1pg13.5,/,' qstar: ',1pg13.5,t40,'qstre: ',1pg13.5)

c
c*** before restarting, check to see if this may be a problem child.
c        If the current time is less than SRCSS, this may be a problem
c        simulation.  The scheme is to scan the primary source flux for the
c        rest of time.  If the flux is never higher, disable the blanket
c        restart feature in NOBL with the logical flag REFLAG, and
c        force the source to use NOBL.
c
         IF(QSTRE .Gt. QSTAR .and. time0.ge.srcss) then
                goto 100

         else if(QSTRE .GT. QSTAR) then

                do iii=1,igen
                if(ptime(iii) .eq. poundn) goto 100
                if(time0 .lt. ptime(iii)) goto 101
                enddo
                goto 100

101             do ii=iii,igen
                if(et(ii) .eq. 0.D0) goto 102
                flux = et(ii)/(pi*r1t(ii)**2)
                if(flux .gt. qstre) goto 100
                enddo

102             reflag = .false.
                write(lunlog,1146)
1146            FORMAT(' Forcing NOBL...')
        endif
```

```
   C.........................................................
   C
   C       SOURCE INTEGRATION -- NO GAS BLANKET
   C
     105   continue
           WRITE(lunlog,1147)
    1147   FORMAT(5X,'Source calculation - No Gas blanket')
   C
           CALL NOBL(timeout, reflag)
   c
           if(check3) then         ! restart blanket calculation
                   time0 = timeout
                   goto 100
           end if
   C
   C
     110   RMAX = 1.01*RMAX        ! GUARANTEE A GOOD VALUE
           aleph = 0.
           if(u0 .ne. 0.) ALEPH = U0/GAMMAF*(SZM/Z0)**ALPHA
        $          /(SQRTPI/2.*RM +RMAX)**(ALPHA/ALPHA1)/alpha1
   C
   c
           rewind (unit=9)
           opnrup1 = opnrup1(1:nchar) // scl(1:4)
           open(unit=8,name=opnrup1, type='new',
        $  carriagecontrol='fortran', recordtype='variable')
   c
           call head(gmass0)
           call crfg(table,ntab,crfger)
           call head(gmass0)
   c
           CLOSE(UNIT=9)
           close(unit=8)
   C
           opnrup1 = opnrup1(1:nchar) // tr2(1:4)
           CALL TRANS(OPNRUP1)
   C
   C*** CALCULATE EXECUTION TIME
   C
           tt1 = t1
           t1 = Secnds(tT1)/60.
           WRITE(lunlog,2000) TSRC
           WRITE(lunlog,2010) T1
    2000   FORMAT(1X,'BEGAN AT ',A24)
    2010   FORMAT(5X,' ***** ELAPSED TIME ***** ',1pg13.5,' min ')
   C
           STOP
           END
####
```

```
C.....................................................
C
C
C       DIMENSIONS/DECLARATIONS for DEGADIS2
C
        include 'sys$degadis:DEGADIS1.dec/list'
C
C   MAXNOB IS THE MAXIMUM NUMBER OF OBSERVERS ALLOWED.
C
        parameter(       maxnob = 50,
     $                   RT2= 1.41 421 3562D0,    ! sqrt(2.0)
     $                   sqpio2= 1.25 331 4137D0)       ! sqrt(pi/2.)
C
####
```

```
      PROGRAM DEGADIS2
C
C***********************************************************************
C***********************************************************************
C***********************************************************************
C
C       Program description:
C
C       DEGADIS2 performs the downwind dispersion portion of the calculation
C       for each of several observers released successively over the transient
C       gas source described by DEGADIS1.
C
C
C       Program usage:
C
C       Consult Volume III of the Final Report to U. S. Coast Guard
C       contract DT-CG-23-80-C-20029 entitled "Development of an
C       Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures".
C
C       J. A. Havens
C       T. O. Spicer
C
C       University of Arkansas
C       Department of Chemical Engineering
C       Fayetteville, AR 72701
C
C       April 1985
C
C
C       This project was sponsored by the U. S. Coast Guard and the Gas
C       Research Institute under contract DT-CG-23-80-C-20029.
C
C
C       Disclaimer:
C
C       This computer code material was prepared by the University of
C       Arkansas as an account of work sponsored by the U. S. Coast Guard
C       and the Gas Research Institute.  Neither the University of Arkansas,
C       nor any person acting on its behalf:
C
C       a.  Makes any warranty or representation, express or implied,
C           with respect to the accuracy, completeness, or usefulness
C           of the information contained in this computer code material,
C           or that the use of any apparatus, method, numerical model,
C           or process disclosed in this computer code material may not
C           infringe privately owned rights; or
C
C       b.  Assumes any liability with respect to the use of, or for
C           damages resulting from the use of, any information,
C           apparatus, method, or process disclosed in this computer
C           code material.
```

```
C
C
C***************************************************************************
C***************************************************************************
C***************************************************************************
C
C
C
c
c
C
C      DIMENSIONS/DECLARATIONS
C
C


       Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

       include 'sys$degadis:DEGADIS2.dec/list'
       include '($ssdef)'
c
       COMMON
    $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
    $ srcwa(2,maxl),srcenth(2,maxl)
    $/SSCON/ NREC(maxnob,2),TO(maxnob),XV(maxnob)
    $/TITL/ TITLE
    $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
    $       PFRACV(igen), PENTH(igen), PRHO(igen)
    $/GEN2/ DEN(5,igen)
    $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
    $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
    $ gas_ufl,gas_lfl,gas_zsp,gas_name
    $/ITI/ T1,TINP,TSRC,TOBS,TSRT
    $/ERROR/SYOER,ERRO,SZOER,WTAIO,WTQOO,WTSZO,ERRP,SMXP,
    $ WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WTDHG
    $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
    $ humsrc
    $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
    $/STP/ STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
    $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
    $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
    $/NEND/ POUNDN,POUND
    $/ALP/ ALPHA,alpha1
    $/phicom/ iphifl,dellay
    $/sprd_con/ ce, delrhomin
    $/COM_SURF/ HTCUT
    $/STOPIT/ TSTOP
    $/CNOBS/ NOBS
    ./oomsin/ oodist,avtime
c
     character*80 TITLE(4)
c
```

```
      character*4 pound
      character*24 TINP,TSRC,TOBS,TSRT
      character*3 gas_name
c
      real*4 tt1
      REAL*8 K,ML,L
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      character*4 TR2,ER2,PSD,TR3, obs
      character OPNRUP(40)
      character*40 OPNRUP1
      equivalence (opnrup(1),opnrup1)
C
C..........................................................
C
C     DATA
C
      DATA TSTOP/0./
      DATA POUND/'//  '/,POUNDN/-1.E-20/
C
      DATA TIME0/0./,NDIM/0/
      DATA RADG/maxl2*0./,QSTR/maxl2*0./,srcden/maxl2*0./
      DATA NREC/maxnob*0,maxnob*0/,T0/maxnob*0./,XV/maxnob*0./
C
      DATA TR2/'.TR2'/,ER2/'.ER2'/
      DATA PSD/'.PSD'/,TR3/'.TR3'/, obs/'.OBS'/
C
C.........................................................
C
C     MAIN
C
      T1 = SECNDS(0.)
      istat = lib$date_TIME(TOBS)
      if(istat .ne. ss$_normal) stop'lib$date_time failure'
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
c     WRITE(5,1130)
c1130 FORMAT(' Enter the file name being used for this run: ',$)
      read(5,1130) nchar,opnrup
 1130 format(q,40a1)
C
      opnrup1 = OPNRUP1(1:nchar) // er2(1:4)
      CALL ESTRT2(OPNRUP1)
C
C*** GET THE COMMON VARIABLES CARRIED FROM DEGADIS1
C
      opnrup1 = OPNRUP1(1:nchar) // tr2(1:4)
      CALL STRT2(OPNRUP1,H_masrte)
C
C
```

```
C..............................................................
C
C       PSEUDO STEADY STATE CALCULATIONS
C       INTEGRATION IN SUBROUTINE SUPERVISOR
C
        opnrup1 = OPNRUP1(1:nchar) // psd(1:4)
        OPEN(UNIT=9,TYPE='NEW',NAME=OPNRUP1,
     $  carriagecontrol='list',
     $  recordtype='variable')
c
        opnrup1 = OPNRUP1(1:nchar) // obs(1:4)
        OPEN( UNIT=12, TYPE='NEW',NAME=OPNRUP1,
     $  carriagecontrol='list',
     $  recordtype='variable')
C
        CALL SSSUP(H_masrte)
C
C.................................................................
C
        CLOSE(UNIT=9)
c
c
        call setden(1.D0,0.D0,H_masrte) ! adiabatic mixing w/ pure stuff
c
C
        opnrup1 = OPNRUP1(1:nchar) // tr3(1:4)
        CALL TRANS(OPNRUP1)
C
        tt1 = t1
        T1 = SECNDS(tT1)
        T1 = T1/60.
        WRITE(lunlog,4000) TOBS
        WRITE(lunlog,4010) T1
 4000   FORMAT(3X,'BEGAN AT ',A40)
 4010   FORMAT(3X,'*** ELAPSED TIME *** ',1pG13.5,' min')
C
        STOP
        END
####
```

```
c.......................................................
c
c       declarations for DEGADIS3
c
        include 'sys$degadis:DEGADIS2.dec/list'
c
        parameter (     maxnt=40,
     $                  maxtnob=maxnt*maxnob)
c
####
```

```
      PROGRAM DEGADIS3
C
C***********************************************************************
C***********************************************************************
C***********************************************************************
C
C       Program description:
C
C       DEGADIS3 sorts the downwind dispersion calculation made for each of
C       the several observers in DEGADIS2.  The output concentrations at
C       several given times may then be corrected for along-wind dispersion
C       as desired.
C
C
C       Program usage:
C
C       Consult Volume III of the Final Report to U. S. Coast Guard
C       contract DT-CG-23-80-C-20029 entitled "Development of an
C       Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures".
C
C       J. A. Havens
C       T. O. Spicer
C
C       University of Arkansas
C       Department of Chemical Engineering
C       Fayetteville, AR 72701
C
C       April 1985
C
C
C       This project was sponsored by the U. S. Coast Guard and the Gas
C       Research Institute under contract DT-CG-23-80-C-20029.
C
C
C       Disclaimer:
C
C       This computer code material was prepared by the University of
C       Arkansas as an account of work sponsored by the U. S. Coast Guard
C       and the Gas Research Institute.  Neither the University of Arkansas,
C       nor any person acting on its behalf:
C
C       a.  Makes any warranty or representation, express or implied,
C           with respect to the accuracy, completeness, or usefulness
C           of the information contained in this computer code material,
C           or that the use of any apparatus, method, numerical model,
C           or process disclosed in this computer code material may not
C           infringe privately owned rights; or
C
C       b.  Assumes any liability with respect to the use of, or for
C           damages resulting from the use of, any information,
C           apparatus, method, or process disclosed in this computer
```

```
C           code material.
C
C
C***************************************************************************
C***************************************************************************
C***************************************************************************
C
c
C
C


      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS3.dec/list'
      include '($ssdef)'
C
C*** MINIMUM DIMENSION ON TABLE IS 6 * MAXNOB + 1
C
      parameter (ntab0=10*maxnob+1)
C
      COMMON
     $/SORT/ TCA(maxnob,maxnt),TCASTR(maxnob,maxnt),
     $ Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $ Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $ TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $ TDIST0(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/SSCON/ NREC(maxnob,2),T0(maxnob),XV(maxnob)
     $/SORTIN/ TIM(maxnt),NTIM,ISTRT
     $/GEN2/ DEN(5,igen)
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/ERROR/ ERT1,ERDT,ERNTIM
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/CNOBS/ NOBS
     ./oomsin/ oodist,avtime
C
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      REAL*8 ML,K
      DIMENSION TABLE(ntab0)
c
      character*24 tsrc,tinp,tobs,tsrt
C
      character*3 gas_name
```

```
        character*4 TR3,PSD,Er3,SR3,Tr4
        character*40 opnrup1
        character opnrup(40)
C
        EQUIVALENCE (OPNRUP(1),opnrup1)
C..........................................................
C
C       DATA
C
        DATA POUNDN/-1.E-20/,POUND/'//  '/
        DATA TCA/maxtnob*0./,TCASTR/maxtnob*0./,TSY/maxtnob*0./
        data TSZ/maxtnob*0./,KSUB/maxnt*0/
        DATA TB/maxtnob*0./,TDIST0/maxtnob*0./,TDIST/maxtnob*0./
C
        DATA TR3/'.TR3'/,PSD/'.PSD'/
        DATA Er3/'.Er3'/,SR3/'.SR3'/,Tr4/'.Tr4'/
C
C*** UNITS
C*** 8 -- OUTPUT TO A PRINT FILE
C*** 9 -- I/O WITH DISK
C
        T1 = SECNDS(0.)
        istat = lib$date_time(tsrt)
        if(istat .ne. ss$_normal) stop'lib$date_time failure'
C
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
c       WRITE(5,1130)
c1130   FORMAT(' Enter the file name used for this run: ',$)
        read(5,1130) nchar,opnrup
 1130   format(q,40a1)
C
C*** GET THE VERSION NUMBER
C
c 100   WRITE(5,1140)
c1140   FORMAT(' Enter the version number (between 00 and 99) for',
c    $' this sort: ',$)
c       CALL GTLIN(DUMMY)
c       NCAR = LEN(DUMMY)
c       IF(NCAR .EQ. 0) GO TO 110
C
c       IF(IVERIF(DUMMY,STRING) .NE. 0) GO TO 110
c       IF(NCAR-2) 130,140,120
C
c 110   WRITE(5,1150)
c1150   FORMAT(' ?DEGADIS3? - Invalid character for version number')
c       GO TO 100
C
c 120   WRITE(5,1160)
c1160   FORMAT(' ?DEGADIS3? - Too many characters in the version number')
```

```
c       GO TO 100
C
c 130   DOT(1) = "060
c       DOT(2) = DUMMY(1)
c       GO TO 150
C
c 140   DOT(1) = DUMMY(1)
c       DOT(2) = DUMMY(2)
C
c 150   CONTINUE
c       CALL CONCAT(Er3,DOT,Er3)
c       CALL CONCAT(Sr3,DOT,Sr3)
c       CALL CONCAT(Tr4,DOT,Tr4)
C
C*** NOW, REPLACE THE FILE NAME IN OPNRUP
C
c       CALL SCOPY(BFILE,OPNRUP)
C
C*** THATS IT
C
        opnrup1 = opnrup1(1:nchar) // tr3(1:4)
        CALL STRT3(OPNRUP1)
C
        opnrup1 = opnrup1(1:nchar) // er3(1:4)
        CALL ESTRT3(OPNRUP1)
C
        opnrup1 = opnrup1(1:nchar) // psd(1:4)
        OPEN(UNIT=9,NAME=OPNRUP1,TYPE='OLD')
C
C..............................................................
C
C       TIME SORT SUPERVISOR -- CALCULATE DOWNWIND DISPERSION CORRECTION
C
        CALL SORTS(TABLE)
C
        CLOSE(UNIT=9)
C
C..............................................................
C
C       OUTPUT SORTED PARAMETERS
C
        opnrup1 = opnrup1(1:nchar) // SR3(1:4)
        CALL SRTOUT(OPNRUP1, table)
C
        opnrup1 = opnrup1(1:nchar) // tr4(1:4)
        CALL TRANS(OPNRUP1)
C
        STOP
        END
####
```

```
c..........................................................
c
c       declarations for DEGADIS4
c
        include 'sys$degadis:DEGADIS3.dec/list'
c
        parameter (ndos=10)
c
####
```

```
      PROGRAM DEGADIS4
C
C*******************************************************************************
C*******************************************************************************
C*******************************************************************************
C
C       Program description:
C
C       DEGADIS4 sorts the downwind dispersion calculation made for each of
C       the several observers in DEGADIS2.  The output concentrations at
C       several given times may then be corrected for along-wind dispersion
C       as desired.  DEGADIS4 also outputs the concentration as a function
C       of time at a given position.
C
C
C       Program usage:
C
C       Consult Volume III of the Final Report to U. S. Coast Guard
C       contract DT-CG-23-80-C-20029 entitled "Development of an
C       Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures".
C
C       J. A. Havens
C       T. O. Spicer
C
C       University of Arkansas
C       Department of Chemical Engineering
C       Fayetteville, AR 72701
C
C       April 1985
C
C
C       This project was sponsored by the U. S. Coast Guard and the Gas
C       Research Institute under contract DT-CG-23-80-C-20029.
C
C
C       Disclaimer:
C
C       This computer code material was prepared by the University of
C       Arkansas as an account of work sponsored by the U. S. Coast Guard
C       and the Gas Research Institute.  Neither the University of Arkansas,
C       nor any person acting on its behalf:
C
C       a.  Makes any warranty or representation, express or implied,
C           with respect to the accuracy, completeness, or usefulness
C           of the information contained in this computer code material,
C           or that the use of any apparatus, method, numerical model,
C           or process disclosed in this computer code material may not
C           infringe privately owned rights; or
C
C       b.  Assumes any liability with respect to the use of, or for
C           damages resulting from the use of, any information,
C           apparatus, method, or process disclosed in this computer
C           code material.
C
C
C***************************************************************************
C***************************************************************************
C***************************************************************************
C
c
C
C

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADIS4.dec/list'
        include '($ssdef)'
C
C*** MINIMUM DIMENSION ON TABLE IS 6 * MAXNOB + 1
C
        parameter (ntab0=10*maxnob+1)
C
        COMMON
     $/SORT/ TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
     $  Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $  Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $  TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $  TDIST0(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/SSCON/ NREC(maxnob,2),T0(maxnob),XV(maxnob)
     $/cdos/ idos, dosdisx(ndos), dosdis(4,2,ndos)
     $/SORTIN/ TIM(maxnt),NTIM,ISTRT
     $/GEN2/ DEN(5,igen)
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/ERROR/ ERT1,ERDT,ERNTIM
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/CNOBS/ NOBS
C
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      REAL*8 ML,K
      DIMENSION TABLE(ntab0)
c
      character*24 tsrc,tinp,tobs,tsrt
C
```

```
      character*3 gas_name
      character*4 TR3,PSD,Er3,SR4,Tr4
      character*40 opnrup1
      character opnrup(40)
C
      EQUIVALENCE (OPNRUP(1),opnrup1)
C.........................................................
C
C     DATA
C
      DATA POUNDN/-1.E-20/,POUND/'//  '/
      DATA TCc/maxtnob*0./,TCcSTR/maxtnob*0./,TSY/maxtnob*0./
      data TSZ/maxtnob*0./,KSUB/maxnt*0/
      DATA TB/maxtnob*0./,TDIST0/maxtnob*0./,TDIST/maxtnob*0./
C
      DATA TR3/'.TR3'/,PSD/'.PSD'/
      DATA Er3/'.Er3'/,SR4/'.SR4'/,Tr4/'.Tr4'/
C
C*** UNITS
C*** 8 -- OUTPUT TO A PRINT FILE
C*** 9 -- I/O WITH DISK
C
      T1 = SECNDS(0.)
      istat = lib$date_time(tsrt)
      if(istat .ne. ss$_normal) stop'lib$date_time failure'
C
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
      WRITE(lunlog,1130)
1130  FORMAT(' Enter the file name used for this run: ',$)
      read(5,1131) nchar,opnrup
 1131 format(q,40a1)
C
C
      opnrup1 = opnrup1(1:nchar) // tr3(1:4)
      CALL STRT3(OPNRUP1)
C
      opnrup1 = opnrup1(1:nchar) // er3(1:4)
      CALL ESTRT3(OPNRUP1)
C
c
c.....................................
c
c     input the position to calculate the concentration histograms
c
      write(6,*) 'Enter the number of downwind distances desired:'
      write(6,*) ' max of ',ndos,
     1      ' downwind distances; 4 positions at each distance'
      read(5,*) jdos
      if(jdos .gt. ndos) jdos=ndos
```

```
        do ii=1,jdos
        write(6,*) ' '
        write(6,*) 'enter the x coordinate:'
        read(5,*) dosdisx(ii)
                write(6,*)
        1       '  enter the y and z coordinate pairs at this distance:'
                do ij=1,4
                read(5,*) (dosdis(ij,jj,ii),jj=1,2)
                if(dosdis(ij,2,ii).le.0. .and.
        1               dosdis(ij,1,ii).le.0.) goto 100
                enddo
 100    continue
        enddo
c
c
        opnrup1 = opnrup1(1:nchar) // SR4(1:4)
        open(unit=8,name=opnrup1, type='new',carriagecontrol='fortran')

        do idos=1,jdos
C
C.....................................................................
C
C       TIME SORT SUPERVISOR -- CALCULATE DOWNWIND DISPERSION CORRECTION
C
        opnrup1 = opnrup1(1:nchar) // psd(1:4)
        OPEN(UNIT=9,NAME=OPNRUP1,TYPE='OLD')
c
        do ij=1,maxnt
        KSUB(ij) = 0
        do ijk=1,maxnob
        TSZ(ijk,ij) = 0.
        enddo
        enddo
c
c
        CALL SORTS(TABLE)
C
        CLOSE(UNIT=9)
C
C..................................................................
C
C       OUTPUT SORTED PARAMETERS
C
        CALL dosOUT(table)
c
        enddo
c
        close(unit=8)
c
C
```

```
         opnrup1 = opnrup1(1:nchar) // tr4(1:4)
         CALL TRANS(OPNRUP1)
   C
         STOP
         END
####
```

```
C.......................................................................
C
c       declarations for DEGADISIN
C
C

        parameter(      igen= 30,   ! dimension of /gen1/ and /gen2/
     $                  pi= 3.14 159 265 358 24D0)
 c
####
```

```
C***********************************************************************
C***********************************************************************
C***********************************************************************
C
C       PROGRAM DEGADISIN
C
C       Program description:
C
C       DEGADISIN acts as an interactive input module to the programs
C       which make up the DEGADIS model.  The user is guided through a
C       series of questions which supply the model with the necessary
C       input information.
C
C
C       Program usage:
C
C       Consult Volume III of the Final Report to U. S. Coast Guard
C       contract DT-CG-23-80-C-20029 entitled "Development of an
C       Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures".
C
C       J. A. Havens
C       T. O. Spicer
C
C       University of Arkansas
C       Department of Chemical Engineering
C       Fayetteville, AR 72701
C
C       April 1985
C
C
C       This project was sponsored by the U. S. Coast Guard and the Gas
C       Research Institute under contract DT-CG-23-80-C-20029.
C
C
C       Disclaimer:
C
C       This computer code material was prepared by the University of
C       Arkansas as an account of work sponsored by the U. S. Coast Guard
C       and the Gas Research Institute.  Neither the University of Arkansas,
C       nor any person acting on its behalf:
C
C       a.  Makes any warranty or representation, express or implied,
C           with respect to the accuracy, completeness, or usefulness
C           of the information contained in this computer code material,
C           or that the use of any apparatus, method, numerical model,
C           or process disclosed in this computer code material may not
C           infringe privately owned rights; or
C
C       b.  Assumes any liability with respect to the use of, or for
C           damages resulting from the use of, any information,
C           apparatus, method, or process disclosed in this computer
```

```
C           code material.
C
C
C*******************************************************************************
C*******************************************************************************
C*******************************************************************************
C
C
C
C       INITIAL INPUT FOR DEGADIS ROUTINES
c
c       note: this series of programs relies on the system wide
c               logical symbol SYS$DEGADIS which denotes the source
c               and executable code for these images.
C
c
        PROGRAM DEGADISIN

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        include 'SYS$DEGADIS:degadisin.dec'
c
        COMMON
     $/TITL/ TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/NEND/ POUNDN,POUND
     ./comsin/ oodist,avtime
C
        character*80 TITLE(4)
C
        character*3 gas_name
        character*4 pound
        character*24 TSRC,TINP,TOBS,TSRT
C
        REAL*8 ML,K
        LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
c
c check1
c check2=t       cloud type release with no liquid source; SRC1  DEGADIS1
c again          local communications in SSSUP                   SSSUP
c check3         local communications between SRC1 and NOBL     DEGADIS1
```

```
c check4=t      steady state simulation
c check5=t      operator sets sort parameters                           DEGADISIN
c                                                                       ESTRT3
      data CHECK1/.false./,CHECK2/.false./,AGAIN/.false./
      data CHECK3/.false./,CHECK4/.false./,CHECK5/.false./
C
      character*100 OPNRUP
      character OPNRUP1(100)
      equivalence (opnrup1(1),opnrup)
      character*4 INP,er1,er2,er3,com,scl,sr3,lis
      character*4 dummy
      character*3 plus
      character*2 con
      DATA POUND/'//  '/,POUNDN/-1.E-20/
C
      DATA PTIME/igen*0.D0/
      DATA ET/igen*0.D0/,R1T/igen*0.D0/
      DATA PWC/igen*0.D0/,PTEMP/igen*0.D0/
      DATA PFRACV/igen*0.D0/,PENTH/igen*0.D0/
      DATA PRHO/igen*0.D0/
      data DEN/igen*0.,igen*0.,igen*0.,igen*0.,igen*0./
      DATA INP/'.INP'/,er1/'.er1'/,er2/'.er2'/,er3/'.er3'/
      data scl/'.scl'/,sr3/'.sr3'/,lis/'.lis'/
      data com/'.com'/
      data plus/' + '/,con/' -'/
c
C*** GET THE FILE NAME TO BE USED BY ALL OF THE ROUTINES
C
      WRITE(6,800)
      WRITE(6,810)
      READ(5,820) NCHAR,opnrup
      opnrup = opnrup(1:nchar) // inp(1:4)
C
C*** NOW GET THE REST OF THE DESIRED INFORMATION
C
      CALL IOT(OPNRUP)
      WRITE(6,1000)
      if(check4) then
      write(6,1001)   ! continuous
      else
              if(u0 .eq. 0.) then
                      write(6,1009)
              else
                      WRITE(6,1002)   ! transient
              endif
      endif
      write(6,1010)
C
C*** FORMATS
C
  800   FORMAT(//,16X,'DEnse GAs DISpersion Model input module.')
```

```
 810  FORMAT(/,' Enter the simulation name',
     $' : [DIR]RUNNAME ',$)
 820  FORMAT(Q,A40)
1000  FORMAT(' ',/,
     $' In addition to the information just obtained,',
     $' DEGADIS',/,' requires a series of numerical parameter',
     $' files which use',/,' the',
     $' same name as [DIR]RUNNAME given above. ',//,
     $'    For convenience, example parameter files are included for',/,
     $' each step.  They are:')
1001  FORMAT(10X,'EXAMPLE.ER1 and',/,10X,'EXAMPLE.ER2')
1002  format(10X,'EXAMPLE.ER1,',/,10X,'EXAMPLE.ER2, and',/,
     $10X,'EXAMPLE.ER3')
1009  format(10x,'EXAMPLE.ER1')
1010  format('  Note that each of',
     $' these files can be edited during the course of the',/,
     $' simulation if a parameter proves to be out of specification.',/)
c
c
      write(6,1200)
 1200 format(' Do you want a command file to be generated to execute',
     $' the procedure? <Y or n> ',$)
      REad(5,1210) dummy
 1210 format(a4)
      if(dummy.eq.'n' .or. dummy.eq.'N') goto 3000
      opnrup = opnrup(1:nchar) // com(1:4)
      write(6,1220) opnrup
 1220 format(' The command file will be generated under the file',
     $' name:',/,10x,a40)
c
      open(unit=8,name=opnrup,type='new',
     $ carriagecontrol='list',recordtype='variable')
c
      opnrup = opnrup(1:nchar) // er1(1:4)
c
      write(8,1250) (opnrup1(i),i=1,nchar+4)
 1250 format('$ copy/log SYS$DEGADIS:example.er1 ',40a1)
      IF(u0 .eq. 0.) then
              write(8,1280)
              write(8,1290) (opnrup1(i),i=1,nchar)
              goto 1340
              endif
      opnrup = opnrup(1:nchar) // er2(1:4)
c
      write(8,1260) (opnrup1(i),i=1,nchar+4)
 1260 format('$ copy/log SYS$DEGADIS:example.er2 ',40a1)
      opnrup = opnrup(1:nchar) // er3(1:4)
c
      if(.not.check4) then            ! transient
c
              write(8,1270) (opnrup1(i),i=1,nchar+4)
 1270           format('$ copy/log SYS$DEGADIS:example.er3 ',40a1)
c
                write(8,1280)
 1280           format('$ run SYS$DEGADIS:DEGADIS1')
                write(8,1290) (opnrup1(i),i=1,nchar)
 1290           format(40a1)
                write(8,1300)
 1300           format('$ run SYS$DEGADIS:DEGADIS2')
                write(8,1290) (opnrup1(i),i=1,nchar)
                write(8,1320)
 1320           format('$ run SYS$DEGADIS:DEGADIS3')
                write(8,1290) (opnrup1(i),i=1,nchar)
c
                opnrup = opnrup(1:nchar) // scl(1:4) //
        1       plus(1:3) // opnrup(1:nchar) // sr3(1:4) // con(1:2)
                write(8,1370) (opnrup1(i),i=1,2*nchar+13)
 1370           format('$ copy/log ',100a1)
c
                opnrup = opnrup(1:nchar) // lis(1:4)
                write(8,1390) (opnrup1(i),i=1,nchar+4)
 1390           format('  ',40a1)
          else
                write(8,1280)
                write(8,1290) (opnrup1(i),i=1,nchar)
c
                write(8,1330)
 1330           format('$ run SYS$DEGADIS:SDEGADIS2')
                write(8,1290) (opnrup1(i),i=1,nchar)
c
                opnrup = opnrup(1:nchar) // scl(1:4) //
        1       plus(1:3) // opnrup(1:nchar) // sr3(1:4) // con(1:2)
                write(8,1370) (opnrup1(i),i=1,2*nchar+13)
                opnrup = opnrup(1:nchar) // lis(1:4)
                write(8,1390) (opnrup1(i),i=1,nchar+4)
          endif
c
 1340     close(unit=8)
          write(6,1350)
 1350     format(/,' Do you wish to initiate this procedure? ',
     $' <y or N> ',$)
          REad(5,1210) dummy
          if(dummy.eq.'y' .or. dummy.eq.'Y') goto 2000
          goto 3000
 2000     opnrup = '@' // opnrup(1:nchar) // ' '
          istat = lib$do_command(opnrup)
          write(6,2100)
 2100     format(/,' ?DEGADISIN? command file failed to start.')
c
 3000     continue
          CALL EXIT
          END
```

####

```
      SUBROUTINE dosOUT(table)
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS4.dec/list'
C
      COMMON /SORT/TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
     $  Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $  Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $  TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $  TDISTO(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/cdos/ idos, dosdisx(ndos), dosdis(4,2,ndos)
     $/SORTIN/TIM(maxnt),NTIM,ISTRT
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/cometm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/alp/ alpha,alpha1
C
      dimension table(1)
      dimension chist(4,maxnt)
c
      logical cflag,cflag1
c
      character*3 gas_name
C
      if(sigx_flag.eq. 0.) then
            write(8,1102)
      else
            write(8,1104)
            write(8,1105) sigx_coeff,sigx_pow,sigx_min_dist
            endif
c
      cflag = isofl.eq. 1.or. ihtfl.eq. 0
      cflag1= isofl.eq.1
      if(cflag) then
            call adiabat(2,wc,wa,gas_lfl,ya,cc_lfl,r,w,t,tt)
            call adiabat(2,wc,wa,gas_ufl,ya,cc_ufl,r,w,t,tt)
            endif
C
      WRITE(8,1110) dosdisx(idos)
c
      WRITE(8,1119)
      WRITE(8,1119)
      if(cflag1) then
            WRITE(8,1116) gas_zsp,(100.*gas_lfl),(100.*gas_ufl)
            WRITE(8,1118)
      else
            WRITE(8,1115) gas_zsp,(100.*gas_lfl),(100.*gas_ufl)
```

```
               WRITE(8,1117)
               endif
       WRITE(8,1119)
       ip = 0
c
       DO 110 I=ISTRT,NTIM
c
       II = KSUB(I)
c
       j = 1
       disto  = tdist(j,i)
       cco    = tccstr(j,i)
       rhoo   = Trho(j,i)
       yco    = Tyc(j,i)
       tempo  = Ttemp(j,i)
       gammao = Tgamma(j,i)
       bo     = tb(j,i)
       szo    = tsz(j,i)
       syo    = tsy(j,i)
       if( disto .gt. dosdisx(idos) ) then
               write(lunlog,*) ' Extrapolated point for time: ',Tim(i)
       write(8,*) ' Records for ',tim(i),' s are missing - see source'
               goto 110
               endif
c
       DO 120 J=2,II
c
       dist  = tdist(j,i)
       cc    = tccstr(j,i)
       rho   = Trho(j,i)
       yc    = Tyc(j,i)
       temp  = Ttemp(j,i)
       gamma = Tgamma(j,i)
       b     = tb(j,i)
       sz    = tsz(j,i)
       sy    = tsy(j,i)
       blfl  = 0.
       bufl  = 0.

       if( dist .lt. dosdisx(idos) ) goto 119
       dist  = (dosdisx(idos) - disto)/(dist-disto)
       cc    = dist*(cc-cco) + cco
       rho   = dist*(rho-rhoo) + rhoo
       yc    = dist*(yc-yco) + yco
       temp  = dist*(temp-tempo) + tempo
       gamma = dist*(gamma-gammao) + gammao
       b     = dist*(b-bo) + bo
       sz    = dist*(sz-szo) + szo
       sy    = dist*(sy-syo) + syo
c
       if(.not.cflag) then
```

```
              call adiabat(-2,wc,wa,gas_lfl,ya,cc_lfl,r,w,gamma,tt)
              call adiabat(-2,wc,wa,gas_ufl,ya,cc_ufl,r,w,gamma,tt)
              endif
c
c*** calc the concentration time histories
c
       do ij=1,4
              chist(ij,i) = 0.
          if(dosdis(ij,2,idos) .ge. 0.) then
       arg = (dosdis(ij,2,idos)/sz)**alpha1
       if(dosdis(ij,1,idos) .gt. b) arg = arg +
      1                        ((dosdis(ij,1,idos)-b)/sy)**2
             if(arg .lt. 80.) then
                    chist(ij,i) = cc/exp(arg)
       if(cflag) then
             call adiabat(0,xx,aa,ycc,yaa,chist(ij,i),rr,ww,gg,tt)
       else
             call adiabat(-1,xx,aa,ycc,yaa,chist(ij,i),rr,ww,gamma,tt)
       endif
             chist(ij,i) = ycc
             endif
         endif
       enddo
c
c
       arg = (gas_zsp/sz)**alpha1
       if(arg .ge. 80.) then
          if(cflag1) then
             WRITE(8,1120) tim(I),yc,Cc,rho,temp,B,SZ,SY
          else
             WRITE(8,1120) tim(I),yc,Cc,rho,gamma,temp,B,SZ,SY
          endif
          goto 600
       endif
c
       ccz = cc/exp(arg)
       if(ccz .lt. cc_lfl) then
              if(cflag1) then
       WRITE(8,1120) tim(I),yc,Cc,rho,temp,B,SZ,SY
                    else
       WRITE(8,1120) tim(I),yc,Cc,rho,gamma,temp,B,SZ,SY
                    endif
             goto 600
             endif
       arg = -(dlog(cc_lfl/cc) + (gas_zsp/sz)**alpha1)
       blfl = sqrt(arg)*sy + b
c
       if(ccz .lt. cc_ufl) then
              if(cflag1) then
       WRITE(8,1120) tim(I),yc,Cc,rho,temp,B,SZ,SY,blfl
                    else
       WRITE(8,1120) tim(I),yc,Cc,rho,gamma,temp,B,SZ,SY,blfl
                   endif
             goto 600
             endif
       arg = -(dlog(cc_ufl/cc) + (gas_zsp/sz)**alpha1)
       bufl = sqrt(arg)*sy + b
             if(cflag1) then
       WRITE(8,1120) tim(I),yc,Cc,rho,temp,B,SZ,SY,blfl,bufl
                   else
       WRITE(8,1120) tim(I),yc,Cc,rho,gamma,temp,B,SZ,SY,blfl,bufl
                   endif
c
  600  continue
       goto 121
  119  continue
       disto  = tdist(j,i)
       cco    = tccstr(j,i)
       rhoo   = Trho(j,i)
       yco    = Tyc(j,i)
       tempo  = Ttemp(j,i)
       gammao = Tgamma(j,i)
       bo     = tb(j,i)
       szo    = tsz(j,i)
       syo    = tsy(j,i)
  120  CONTINUE
c
  121  ip = ip + 1
       if(ip .eq. 3) then
             ip = 0
             write(8,1119)
             endif
  110  CONTINUE
C
c*** output the concen time histories
c
      write(8,1119)
      write(8,1119)
      iii = 0
      do ij=1,4
        if(dosdis(ij,2,idos).gt.0. .and.
      1            dosdis(ij,1,idos).gt.0.) iii = ij
      enddo

      if(iii .eq. 0) return
      ip = 0

      write(8,2100) ((dosdis(ij,iij,idos),ij=1,4),iij=1,2)
 2100 format(1H0,11x,'Time',11x,4(4x,'Mole fraction at:',5x),/,
      1        1H ,26x,4(4x,'y= '1pg13.5,' m',4x),/,
      1        1H ,11x,'(s)',12x,4(4x,'z= '1pg13.5,' m',4x))
      DO I=ISTRT,NTIM
```

```
          write(8,2200) tim(i),(chist(ij,i),ij=1,iii)
          ip = ip + 1
          if(ip .eq. 3) then
                  ip = 0
                  write(8,1119)
                  endif
          enddo
 2200     format(1h ,5(6x,1pg14.7,6x))
C
C
 1102     format(1H0,5x,'X-Direction correction was NOT applied.')
 1104     format(1H0,5x,'X-Direction correction was applied.')
 1105     format(1h ,5x,5x,'Coefficient:      ',1pg13.5,/,
     1           1h ,5x,5x,'Power:            ',1pg13.5,/,
     1           1h ,5x,5x,'Minimum Distance: ',1pg13.5' m')
 1110     FORMAT(1H0,5X,'Centerline values for the position -->',/,
     1           ' x: ',1pg13.5,' m',/)
 1115     FORMAT(1H0,1X,'  Time  ',2x,3x,'Mole',3x,
     1           'Concentration',1x,'Density',2x,3x,'Gamma',3x,
     1           'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
     1           'Width at z=',0pf6.2,' m to:',/,1x,11x,1x'Fraction',2x,
     1           11x,11x,11x,11x,3x,'Width',3x,11x,9x,
     1           2(1pg9.3,'mole%',1x))
 1116     FORMAT(1H0,1X,'  Time  ',2x,3x,'Mole',3x,
     1           'Concentration',1x,'Density',2x,
     1           'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
     1           'Width at z=',0pf6.2,' m to:',/,1x,11x,1x'Fraction',2x,
     1           11x,11x,11x,3x,'Width',3x,11x,9x,
     1           2(1pg9.3,'mole%',1x))
 1117     FORMAT(1H ,4X,'(s)',4x,11x,
     1           2(1X,'(kg/m**3)',1x),11x,4x,'(K)',
     1           5(8X,'(m)'))
 1118     FORMAT(1H ,4X,'(s)',4x,11x,
     1           2(1X,'(kg/m**3)',1x),4x,'(K)',
     1           5(8X,'(m)'))
 1119     FORMAT(1H )
 1120     FORMAT(1H ,3(1X,1PG9.3,1X),2x,0pf7.4,2x,1X,1PG10.3,1X,
     1           6(1X,1PG9.3,1X))
C
          RETURN
          END
####
```

```
C.............................................................
C
C       ROUTINE TO GET RUN PARAMETERS FROM A FILE
C
        SUBROUTINE ESTRT1(OPNRUP)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        include 'sys$degadis:DEGADIS1.dec'
C
        parameter(      iend= 22,
        1               iend1= iend+1,
        1               iiiend= 2,
        1               iiiend1= iiiend+1,
        1               iiend= 2,
        1               iiend1= iiend+1,
        1               jend= 4,
        1               jend1= jend+1,
        1               mend= 5,
        1               mend1= mend+1)
C
C       BLOCK COMMON
C
        COMMON
     $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/vucom/ vua,vub,vuc,vud,vudelta,vuflag
     $/szfc/ szstp0,szerr,szstpmx,szsz0
     $/alphcom/ ialpfl,alpco
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
C
        EQUIVALENCE
     $(RLBUF(1),STPIN),   !MAIN - RKGST - INITIAL STEP SIZE
     $(RLBUF(2),ERBND),   !MAIN - RKGST - ERROR BOUND
     $(RLBUF(3),STPMX),   !MAIN - RKGST - MAXIMUM STEP SIZE
     $(RLBUF(4),WTRG),    !MAIN - RKGST - WEIGHT FOR RG
     $(RLBUF(5),WTtm),    !MAIN - RKGST - WEIGHT FOR Total mass
     $(RLBUF(6),WTya),    !MAIN - RKGST - WEIGHT FOR Ya
     $(RLBUF(7),WTyc),    !MAIN - RKGST - WEIGHT FOR Yc
     $(RLBUF(8),WTeb),    !MAIN - RKGST - WEIGHT FOR Energy Balance
     $(RLBUF(9),WTmb),    !MAIN - RKGST - WEIGHT FOR Momentum Balance
     $(RLBUF(10),WTuh),   !MAIN - RKGST - WEIGHT FOR Ueff*Heff
     $(RLBUF(11),XLI),    !ALPH - LOWER LIMIT OF SEARCH FOR ALPHA
     $(RLBUF(12),XRI),    !ALPH - UPPER LIMIT OF SEARCH FOR ALPHA
     $(RLBUF(13),EPS)     !ALPH - ERROR BOUND USED BY "RTMI"
        equivalence
     $(RLBUF(14),ZLOW),   !ALPHI - BOTTOM HEIGHT FOR FIT OF ALPHA
     $(RLBUF(15),STPINZ), !ALPHI - INITIAL RKGST STEP <0.
     $(RLBUF(16),ERBNDZ), !ALPHI - ERROR BOUND FOR RKGST
     $(RLBUF(17),STPMXZ), !ALPHI - MAXIMUM STEP FOR RKGST
     $(RLBUF(18),SRCOER), !SRC1O - OUTPUT error criteria
     $(RLBUF(19),SRCSS),  !SRC1O - min time for Steady;4*STPMX
     $(RLBUF(20),SRCcut), !SRC1O - min height for blanket
     $(RLBUF(21),htcut),  !SRC1 - min height for blanket heat transfer
     $(RLBUF(iend),ERNOBL), !NOBL - CONVERGENCE CRITERIA
     $(RLBUFi(1),crfger), !CRFG - Error criteria for building tables
     $(RLBUFi(iiiend),epsilon)!SRC1 - Coefficient in Air entrainment
c
        equivalence
     $(rlbufa(1),ce),   !SRC1 - Coefficient gravity slumping EQ
     $(RLBUFa(iiend),delrhomin) ! stop spread for delrho<delrhomin
c
        equivalence
     $(rlbuf1(1),szstp0),  ! SZF - Initial step size
     $(rlbuf1(2),szerr),   ! SZF - Error criteria
     $(rlbuf1(3),szstpmx), ! SZF - Maximum step size
     $(rlbuf1(4),szsz0)    ! SZF - Initial value of rho*dellay*UHeff
C
c
        equivalence
     $(rlbuf4(1),vua),  ! Constant Av in SRC1
     $(rlbuf4(2),vub),  ! Constant Bv in SRC1
     $(rlbuf4(3),vuc),  ! Constant Ev in SRC1
     $(rlbuf4(4),vud),  ! Constant Dv in SRC1
     $(rlbuf4(5),vudelta)       ! Constant DELTAv in SRC1
c
        character*40 OPNRUP
        character DUMMY(1:132)
        DIMENSION RLBUF(iend), rlbufi(iiiend), rlbufa(iiend)
        dimension rlbuf1(jend)
        dimension rlbuf4(mend)
c
        logical vuflag
C
        OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD',err=2000)
C
C*** READ A LINE AND DETERMINE ITS PURPOSE
C
        I = 1
 100    CONTINUE
        READ(9,1000,END=350) NCHAR,DUMMY
        IF(DUMMY(1) .EQ. '!') GO TO 100
        DECODE(20,1010,DUMMY,ERR=400) RLBUF(I)
        I = I + 1
        if(i .eq. iend1) goto 110
        GO TO 100

 110    CONTINUE
        READ(9,1000,END=350) NCHAR,DUMMY
```

```
      IF(DUMMY(1) .EQ. '!') GO TO 110
      DECODE(20,1010,DUMMY,ERR=400) ptnobl
      NOBLPT = INT(PTNOBL)

      I = 1
  120 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 120
      DECODE(20,1010,DUMMY,ERR=400) RLBUFi(I)
      I = I + 1
      if(i .eq. iiend1) goto 140
      GO TO 120
C
C*** READ A LINE AND DETERMINE ITS PURPOSE for /sprd_con/
C
  140 I = 1
  150 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 150
      DECODE(20,1010,DUMMY,ERR=400) RLBUFa(I)
      I = I + 1
      if(i .eq. iiend1) goto 190
      GO TO 150
C
C*** READ A LINE AND DETERMINE ITS PURPOSE to fill szfc
C
  190 I = 1
  200 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 200
      DECODE(20,1010,DUMMY,ERR=400) RLBUF1(I)
      I = I + 1
      if(i .eq. jend1) goto 230
      GO TO 200
c
C*** READ A LINE AND DETERMINE ITS PURPOSE to fill /alphcom/
C
  230 I = 1
  240 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 240
      DECODE(20,1010,DUMMY,ERR=400) Ralpfl

      ialpfl = int(ralpfl)

  250 CONTINUE
      READ(9,1000,END=350) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 250
      DECODE(20,1010,DUMMY,ERR=400) alpco
c
C*** READ A LINE AND DETERMINE ITS PURPOSE to fill /phicom/
```

```
C
  260   I = 1
  270   CONTINUE
        READ(9,1000,END=350) NCHAR,DUMMY
        IF(DUMMY(1) .EQ. '!') GO TO 270
        DECODE(20,1010,DUMMY,ERR=400) Rphifl

        iphifl = int(rphifl)

  275   CONTINUE
        READ(9,1000,END=350) NCHAR,DUMMY
        IF(DUMMY(1) .EQ. '!') GO TO 275
        DECODE(20,1010,DUMMY,ERR=400) dellay
c
C*** READ A LINE AND DETERMINE ITS PURPOSE to fill /vucom/
C
  280   I = 1
  290   CONTINUE
        READ(9,1000,END=350) NCHAR,DUMMY
        IF(DUMMY(1) .EQ. '!') GO TO 290
        DECODE(20,1010,DUMMY,ERR=400) RLBUF4(I)
        I = I + 1
        if(i .eq. mend1) goto 300
        GO TO 290
c
C*** EXIT THE PROCEEDINGS
C
  300   CONTINUE
        CLOSE(UNIT=9)
        RETURN
c
  350   call trap(20)
C
  400   CONTINUE
        CALL trap(21)
C
 1000   FORMAT(Q,132A1)
 1010   FORMAT(10X,G10.4)
C
 2000   call trap(22)
        END
####
```

```
C..............................................................
C
C       ROUTINE TO GET RUN PARAMETERS FROM A FILE
C
        SUBROUTINE ESTRT2(OPNRUP)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        include 'sys$degadis:DEGADIS2.dec/list'
c
        parameter (     iendа= 18,
        1               iendа1= ienda+1,
        2               iendb= 7,
        3               iendb1= iendb+1)
c
        common
     $/ERROR/SYOER,ERRO,SZOER,WTAIO,WTQOO,WTSZO,ERRP,SMXP,
     $ WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WTDHG
     $/STP/STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/CNOBS/NOBS
C
        EQUIVALENCE
     $(RLBUF(1),SYOER),   !SSSUP - RKGST - INITIAL SY
     $(RLBUF(2),ERRO),    !SSSUP - RKGST(OBS) - ERROR BOUND
     $(RLBUF(3),SZOER),   !SSSUP - RKGST(OBS) - INITIAL SZ
     $(RLBUF(4),WTAIO),   !SSSUP - RKGST(OBS) - WEIGHT FOR AI
     $(RLBUF(5),WTQOO),   !SSSUP - RKGST(OBS) - WEIGHT FOR Q
     $(RLBUF(6),WTSZO),   !SSSUP - RKGST(OBS) - WEIGHT FOR SZ
     $(RLBUF(7),ERRP),    !SSSUP - RKGST(PSS) - ERROR BOUND
     $(RLBUF(8),SMXP),    !SSSUP - RKGST(PSS) - MAXIMUM STEP
     $(RLBUF(9),WTSZP),   !SSSUP - RKGST(PSS) - WEIGHT FOR SZ
     $(RLBUF(10),WTSYP),  !SSSUP - RKGST(PSS) - WEIGHT FOR SY
     $(RLBUF(11),WTBEP),  !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
     $(RLBUF(12),WTDH),   !SSSUP - RKGST(PSS) - WEIGHT FOR DH
     $(RLBUF(13),ERRG),   !SSSUP - RKGST(SSG) - ERROR BOUND
     $(RLBUF(14),SMXG),   !SSSUP - RKGST(SSG) - MAXIMUM STEP SIZE
     $(RLBUF(15),ERTDNF), !TDNF - CONVERGENCE CRITERIA
     $(RLBUF(16),ERTUPF), !TUPF - CONVERGENCE CRITERIA
     $(RLBUF(17),WTruh),  !SSSUP - RKGST(SSG) - WEIGHT FOR RUH
     $(RLBUF(ienda),WTdhg)!SSSUP - RKGST(SSG) - WEIGHT FOR DH
C
        EQUIVALENCE
     $(RLBUF1(1),STPO),   !SSSUP - RKGST(OBS) - INITIAL STEP
     $(RLBUF1(2),STPP),   !SSSUP - RKGST(PSS) - INITIAL STEP
     $(RLBUF1(3),ODLP),   !SSSUP - RKGST(PSS) - RELATIVE OUTPUT DELTA
     $(RLBUF1(4),ODLLP),  !SSSUP - RKGST(PSS) - MAXIMUM DISTANCE TO OUT
     $(RLBUF1(5),STPG),   !SSSUP - RKGST(SSG) - INITIAL STEP
     $(RLBUF1(6),ODLG),   !SSSUP - RKGST(SSG) - RELATIVE OUTPUT DELTA
     $(RLBUF1(iendb),ODLLG)!SSSUP- RKGST(SSG) - MAXIMUM DISTANCE TO OUT
C
```

```
      character*40 OPNRUP
      character dummy(1:132)
      DIMENSION RLBUF(ienda),RLBUF1(iendb)
C
      OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
C*** FIRST, FILL RLBUF
C
C*** READ A LINE AND DETERMINE ITS PURPOSE
C
      I = 1
  100 CONTINUE
      READ(9,1000,END=300) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 100
      DECODE(20,1010,DUMMY,ERR=400) RLBUF(I)
      I = I + 1
      IF(I .EQ. ienda1) GO TO 200
      GO TO 100
C
C*** NOW, FILL RLBUF1
C
  200 I = 1
  210 CONTINUE
      READ(9,1000,END=300) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 210
      DECODE(20,1010,DUMMY,ERR=400) RLBUF1(I)
      I = I + 1
      IF(I .EQ. iendb1) GO TO 260
      GO TO 210
C
C*** NOW, PICK UP NOBS
C
  260 CONTINUE
      READ(9,1000,END=300) NCHAR,DUMMY
      IF(DUMMY(1) .EQ. '!') GO TO 260
      DECODE(20,1010,DUMMY,ERR=400) RBUF
      NOBS = INT(RBUF)
C
C*** EXIT THE PROCEEDINGS
C
      CLOSE(UNIT=9)
      RETURN
C
  300 call trap(20)
  400 CALL trap(21)
C
 1000 FORMAT(Q,132A1)
 1010 FORMAT(10X,G10.4)
      END
####
```

```
C...............................................................
C
C       ROUTINE TO GET RUN PARAMETERS FROM A FILE
C
        SUBROUTINE ESTRT2SS(OPNRUP)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

c
        include 'sys$degadis:DEGadis2.dec/list'
c
        parameter(      ienda= 18,
        1               ienda1= ienda+1,
        2               iendb= 7,
        3               iendb1= iendb+1)
C
        COMMON
     $/ERROR/SYOER,ERRP,SMXP,WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,
     $ WTRUH,WTDHG
     $/STP/STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
C
c
        character*40 OPNRUP
        character DUMMY(1:132)
        DIMENSION RLBUF(ienda),RLBUF1(iendb)
C
        OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
C*** FIRST, FILL RLBUF
C
C*** READ A LINE AND DETERMINE ITS PURPOSE
C
        I = 1
  100   CONTINUE
        READ(9,1000,END=350) NCHAR,DUMMY
        IF(DUMMY(1) .EQ. '!') GO TO 100
        DECODE(20,1010,DUMMY,ERR=400) RLBUF(I)
        I = I + 1
        IF(I.EQ. ienda1) GOTO200
        GO TO 100
C
C*** NOW, FILL RLBUF1
C
  200   i = 1
  210   READ(9,1000,END=350) NCHAR,DUMMY
        IF(DUMMY(1) .EQ. '!') GO TO 210
        DECODE(20,1010,DUMMY,ERR=400) RLBUF1(I)
        I = I + 1
        if(i.eq. iendb1) goto 300
        GO TO 210
C
```

```
C*** EXIT THE PROCEEDINGS
C
  300   CONTINUE
        sy0er = rlbuf(1)  !SSSUP - RKGST - INITIAL SY
        errp  = rlbuf(7)  !SSSUP - RKGST(PSS) - ERROR BOUND
        smxp  = rlbuf(8)  !SSSUP - RKGST(PSS) - MAXIMUM STEP
        wtszp = rlbuf(9)  !SSSUP - RKGST(PSS) - WEIGHT FOR SZ
        wtsyp = rlbuf(10) !SSSUP - RKGST(PSS) - WEIGHT FOR SY
        wtbep = rlbuf(11) !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
        wtDH  = rlbuf(12) !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
        errg  = rlbuf(13) !SSSUP - RKGST(SSG) - ERROR BOUND
        smxg  = rlbuf(14) !SSSUP - RKGST(SSG) - MAXIMUM STEP SIZE
        wtRUH = rlbuf(17) !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
        wtDHG = rlbuf(18) !SSSUP - RKGST(PSS) - WEIGHT FOR BEFF
C
        stpp  = rlbuf1(2) !SSSUP - RKGST(PSS) - INITIAL STEP
        odlp  = rlbuf1(3) !SSSUP - RKGST(PSS) - RELATIVE OUTPUT DELTA
        odllp = rlbuf1(4) !SSSUP - RKGST(PSS) - MAXIMUM DISTANCE TO OUT
        stpg  = rlbuf1(5) !SSSUP - RKGST(SSG) - INITIAL STEP
        odlg  = rlbuf1(6) !SSSUP - RKGST(SSG) - RELATIVE OUTPUT DELTA
        odllg = rlbuf1(7) !SSSUP - RKGST(SSG) - MAXIMUM DISTANCE TO OUT
C
        CLOSE(UNIT=9)
        RETURN
C
  350   call trap(20)   ! premature EOF
c
  400   CALL trap(21)
C
 1000   FORMAT(Q,132A1)
 1010   FORMAT(10X,G10.4)
C
        END
####
```

```
C.....................................................
C
C       ROUTINE TO GET RUN PARAMETERS FROM A FILE
C
        SUBROUTINE ESTRT3(OPNRUP)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
        COMMON
     $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/ERROR/ERT1,ERDT,ERNTIM
C
        EQUIVALENCE
     $(RLBUF(1),ERT1),     !FIRST SORT TIME - USER OPTION
     $(RLBUF(2),ERDT),     !SORT TIME DELTA - USER OPTION
     $(RLBUF(3),ERNTIM)    !NUMBER OF SORT TIMES - USER OPTION
C
        LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
        character DUMMY(1:132)
        character*40 opnrup
        DIMENSION RLBUF(3),RBUF(6)
C
        OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
C*** READ A LINE AND DETERMINE ITS PURPOSE
        I = 1
  100   CONTINUE
        READ(9,1000,END=300) NCHAR,DUMMY
        IF(DUMMY(1) .EQ. '!') GO TO 100
        DECODE(20,1010,DUMMY,ERR=400) RBUF(I)
        I = I + 1
        GO TO 100
C
C*** EXIT THE PROCEEDINGS AND DETERMINE CHECK5
  300   CONTINUE
C
        DO 310 I = 1,3
  310   RLBUF(I) = RBUF(I)
        CHECK5 = .FALSE.                ! IN ORDER FOR FLAG TO WORK
        IF(RBUF(4) .EQ. 1.) CHECK5 = .TRUE.
C
        sigx_flag = rbuf(5)
        CLOSE(UNIT=9)
        RETURN
C
  400   CALL trap(21)
 1000   FORMAT(Q,132A1)
 1010   FORMAT(10X,G10.4)
        END
####
```

```
C
C       ............................................................
C
C       SUBROUTINE GAMMA
C
c.................................................................
c
c       This routine was originally supplied by Digital Equipment
c       Corporation as part of the Scientific Subroutine Package
c       available for RT-11 as part of the Fortran Enhancement
c       Package.  It was upgraded for use in this package.
c
c.............................................................
c
C       PURPOSE
C               COMPUTES THE GAMMA FUNCTION FOR A GIVEN ARGUMENT
C
C       USAGE
C               GF = GAMMA(XX)
C
C       DESCRIPTION OF PARAMETERS
C               XX -THE ARGUMENT FOR THE GAMMA FUNCTION
C
C       IER-RESULTANT ERROR CODE WHERE
C               IER=0  NO ERROR
C               IER=1  XX IS WITHIN .000001 OF BEING A NEGATIVE INTEGER
C               IER=2  XX GT 34.5, OVERFLOW
C               IF IER .NE. 0 PROGRAM TAKES A DIP IN THE POOL!
C
C       REMARKS
C               NONE
C
C       SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C               NONE
C
C       METHOD
C               THE RECURSION RELATION AND POLYNOMIAL APPROXIMATION
C               BY C.HASTINGS,JR., 'APPROXIMATIONS FOR DIGITAL COMPUTERS',
C               PRINCETON UNIVERSITY PRESS, 1955
C
C       MODIFIED TO FUNCTION FORM FROM ORIGINAL SUBROUTINE FORM
C
C     ...................................................................
C
        FUNCTION GAMMA(XX)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        IF(XX-34.5)6,6,4
    4   IER=2
        GAMMA=1.E38
```

```
      GO TO 1000
    6 X=XX
      ERR=1.0E-6
      IER=0
      GAMMA=1.0
      IF(X-2.0)50,50,15
   10 IF(X-2.0)110,110,15
   15 X=X-1.0
      GAMMA=GAMMA*X
      GO TO 10
   50 IF(X-1.0)60,120,110
C
C     SEE IF X IS NEAR NEGATIVE INTEGER OR ZERO
C
   60 IF(X-ERR)62,62,80
   62 Y=FLOAT(INT(X))-X
      IF(ABS(Y)-ERR)130,130,64
   64 IF(1.0-Y-ERR)130,130,70
C
C     X NOT NEAR A NEGATIVE INTEGER OR ZERO
C
   70 IF(X-1.0)80,80,110
   80 GAMMA=GAMMA/X
      X=X+1.0
      GO TO 70
  110 Y=X-1.0
      GY=1.0+Y*(-0.5771017+Y*(+0.9858540+Y*(-0.8764218+Y*
     $(+0.8328212+Y*(-0.5684729+Y*(+0.2548205+Y*(-0.05149930)))))))
      GAMMA=GAMMA*GY
  120 RETURN
  130 IER=1
 1000 CONTINUE
      IF(IER.EQ.1) WRITE(5,1100)
      IF(IER.EQ.2) WRITE(5,1110)
 1100 FORMAT(5X,'?GAMMA?--ARGUMENT LESS THAN 0.000001')
 1110 FORMAT(5X,'?GAMMA?--ARGUMENT GREATER THAN 34.5--OVERFLOW')
      CALL EXIT
      END
####
```

```
C.......................................................................
C
C       SUBROUTINE TO ESTABLISH THE TIME SORT PARAMETERS
C
        SUBROUTINE GETTIM

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

c
        include 'sys$degadis:DEGADIS3.dec/list'
C
        COMMON
     $/SSCON/ NREC(maxnob,2),TO(maxnob),XV(maxnob)
     $/SORTIN/ TIM(maxnt),NTIM,ISTRT
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/ERROR/ ERT1,ERDT,ERNTIM
     $/ALP/ ALPHA,alpha1
     $/CNOBS/ NOBS
C
        LOGICAL CHECK1,CHECK2,CHECK3,CHECK4,CHECK5,AGAIN
C
        DATA T1/0./,DT/0./,TF/0./
C
C*** IF CHECK5 IS SET, GET THE TIME SORT PARAMETERS FROM /ERROR/
C
        IF(.NOT.CHECK5) GO TO 90
C
        T1 = ERT1
        DT = ERDT
        NTIM = INT(ERNTIM)
        GO TO 95
C
C
C*** This subroutine sets the default time sort windows.
C
C*** The first sort time is set for potential low wind speed cases,
C***   while the last sort time is set for potential high wind speed
C***   cases.  The first sort time is taken to be when the first
C***   observer passes through x=RMAX.  The last sort time is taken
C***   to be when the last observer passes through x=6*RMAX.
C***   The default value for the number of sort times is set to 10.
C***   Obviously, these values generate some sort times which will be
C***   useless; hopefully, these values will show the user where to
C***   look on the next go around.
C*** The number of times has been doubled to 20 and the time interval
C***   has been doubled in order to give more information for lower
C***   concentrations (for the toxic gas problem). tos,5mar86
C
 90     CONTINUE
C
```

```
        T1 = TO(1) + (2.*RMAX)**(1./ALPHA1)/ALEPH
        TF = TO(NOBS) + (6.*RMAX)**(1./ALPHA1)/ALEPH
        NTIM = 20                   ! for toxic gas problem
C
c DT = (TF-T1)/FLOAT(NTIM-1)
        DT = 2.*(TF-T1)/FLOAT(NTIM-1)   ! for toxic gas problem
        DT = FLOAT(INT(DT+.5))
C
        IF(DT .GE. 5.) GO TO 95
        DT = 5.
        NTIM = INT((TF - T1)/DT) + 1
C
   95   CONTINUE
C
        T1 = FLOAT(INT(T1))         !MAKE T1 AN INTEGER VALUE
C
        II = max( ntim,2)           ! fill in the lowest two times at least
        DO 100 I = 1,II
        TIM(I) = DT*FLOAT(I-1) + T1
  100   CONTINUE
C
        RETURN
        END
####
```

```
C...........................................................................
C
C       SUBROUTINE TO ESTABLISH THE TIME SORT PARAMETERS
C
        SUBROUTINE GETTIM

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADIS4.dec/list'
C
        COMMON
     $/SSCON/ NREC(maxnob,2),TO(maxnob),XV(maxnob)
     $/cdos/ idos, dosdisx(ndos), dosdis(4,2,ndos)
     $/SORTIN/ TIM(maxnt),NTIM,ISTRT
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/ERROR/ ERT1,ERDT,ERNTIM
     $/ALP/ ALPHA,alpha1
     $/CNOBS/ NOBS
C
        LOGICAL CHECK1,CHECK2,CHECK3,CHECK4,CHECK5,AGAIN
C
        DATA T1/0./,DT/0./,TF/0./
C
C*** This subroutine sets the time sort windows for concentration as
C     a function of time at a given position.
C
        dist = dosdisx(idos)
C
        T1 = ts( TO(1), dist)   ! time first observer crosses dist
        TF = ts( TO(nobs), dist)        ! time last observer crosses dist
        NTIM = 40
C
        DT = (TF-T1)/FLOAT(NTIM-1)
        DT = dmax1( DBLE(INT(DT+.5)), 1.D0)
        NTIM = min( INT((TF - T1)/DT) + 1, 40)
C
        T1 = FLOAT(INT(T1))     !MAKE T1 AN INTEGER VALUE
        write(lunlog,1000) t1, tf, dt, dist
 1000   format('  t1: ',1pg13.5,'  tf: ',1pg13.5,'  dt: ',1pg13.5,
     $' dist:  ',1pg13.5)
C
        II = max( ntim,2)       ! fill in the lowest two times at least
        DO 100 I = 1,II
        TIM(I) = DT*FLOAT(I-1) + T1
  100   CONTINUE
C
        RETURN
        END
####
```

```
      SUBROUTINE HEAD(gmass0)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      include 'sys$degadis:DEGADIS1.dec'
      include '($ssdef)'
c
      COMMON
     $/GEN3/ RADG(2,maxl),QSTR(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/TITL/ TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/ERROR/ STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl,swcl,swal,senl,srhl
     $/phlag/ check1,check2,again,check3,check4,check5
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/alphcom/ ialpfl,alpco
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     ./oomsin/ oodist,avtime
c
      character*80 TITLE(4)
c
      character*4 pound
      character*24 TINP,TSRC,TOBS,TSRT
      character*3 gas_name
      character*1 stabil(6)
      character*24 id
c
      logical check1,check2,again,check3,check4,check5
c
      REAL*8 K,ML
c
      data stabil/'A','B','C','D','E','F'/
      data iparm/0/
C
      if(iparm .eq. 1) goto 190
      WRITE(8,1100)
 1100 FORMAT(1H0,'*************************',9X,'U O A _ D E G A D I S ',
     $2X,'M O D E L ',2X,'O U T P U T ',2X,'- - ',2X,'V E R S I O N ',
     $2X,'2.1',8X,'**************************')
C
C
        WRITE(8,1111)
c
        WRITE(8,1102) tsrc
 1102   FORMAT(1H ,'********************',23X,
     $'********************',1X,a24,1X,
     $'********************',23X,'********************')
C
        WRITE(8,1111)
C
        WRITE(8,1112) TINP
        WRITE(8,1114) TSRC
        IF(tOBS(1:2).NE.'  ' .and. .not.check4) WRITE(8,1116) TOBS
        IF(tOBS(1:2).NE.'  ' .and. check4) WRITE(8,1117) TOBS
        IF(tSRT(1:2) .NE. '  ') WRITE(8,1118) TSRT
 1112   FORMAT(1H ,'Data input on',22X,a24)
 1114   FORMAT(1H ,'Source program run on',14X,a24)
 1116   FORMAT(1H ,'Pseudo Steady-State program run on ',a24)
 1117   FORMAT(1H ,'Steady-State program run on',7x,a24)
 1118   FORMAT(1H ,'Time sort program run on',11X,a24)
        WRITE(8,1111)
C
        WRITE(8,1110)
        WRITE(8,1111)
C
 1110   FORMAT(1H0,10X,'TITLE BLOCK')
 1111   FORMAT(1H )
C
        DO 100 I = 1,4
        WRITE(8,1120) TITLE(I)
  100   CONTINUE
C
 1120   FORMAT(1H ,A80)
C
        WRITE(8,1111)
        WRITE(8,1130) UO
        WRITE(8,1140) ZO
        WRITE(8,1150) ZR
        write(8,1155) stabil(istab)
        if(ml.ne. 0.) then
                WRITE(8,1160) ML
        else
                write(8,1161)
        endif
        write(8,1168) avtime
        WRITE(8,1170) DELTA
        WRITE(8,1180) BETA
        WRITE(8,1190) ALPHA
        WRITE(8,1192) USTAR
        WRITE(8,1194) tamb
        if(isofl.eq.0 .and. ihtfl.ne.0) write(8,1195) tsurf
        WRITE(8,1196) pamb
        WRITE(8,1198) humid
        vaporp = 6.0298e-3* exp(5407.* (1./273.15- 1./tamb)) ! atm
        relhumid = 100.* humid/(0.622*vaporp / (pamb- vaporp))
        write(8,1199) relhumid
C
 1130   FORMAT(1H ,5X,'Wind velocity at reference height ',20X,F6.2,2X,
     $'m/s')
 1140   FORMAT(1H ,5X,'Reference height ',37X,F6.2,2X,'m')
 1150   FORMAT(1H0,5X,'Surface roughness length ',25X,1PG10.3,2X,'m')
 1155   FORMAT(1H0,5X,'Pasquill Stability class ',25X,4x,a1)
 1160   FORMAT(1H0,5X,'Monin-Obukhov length ',29X,1PG10.3,2X,'m')
 1161   FORMAT(1H0,5X,'Monin-Obukhov length ',31X,'infinite')
 1168   FORMAT(1H ,5X,'Gaussian distribution constants ',/,
     .          1H ,5x,8x,9x,'Specified averaging time',10X,F9.2,2X,'s')
 1170   FORMAT(1H ,5X,32x,4X,'Delta',10X,F9.5)
 1180   FORMAT(1H ,5X,32X,4X,' Beta',10X,F9.5)
 1190   FORMAT(1H0,5X,'Wind velocity power law constant',4X,'Alpha',
     $10X,F9.5)
 1192   FORMAT(1H ,5X,'Friction velocity',15X,4X,5X,10X,F9.5,2X,'m/s')
 1194   FORMAT(1H0,5X,'Ambient Temperature ',34X,F6.2,2X,'K')
 1195   FORMAT(1H0,5X,'Surface Temperature ',34X,F6.2,2X,'K')
 1196   FORMAT(1H ,5X,'Ambient Pressure ',37X,F6.3,2X,'atm')
 1198   FORMAT(1H ,5X,'Ambient Absolute Humidity',25X,1PG10.3,2X,
     $'kg/kg BDA')
 1199   FORMAT(1H ,5X,'Ambient Relative Humidity',25X,4X,F6.2,2X,'%')
C
        WRITE(8,1111)
C
        if(isofl .eq. 0) goto 135
        WRITE(8,1200)
        WRITE(8,1205)
        ii = -1
        DO 130 I = 1,igen
        IF(DEN(1,I).gt. 1.) goTO 148
        ii = ii+1
        if(ii.eq. 3) then
                write(8,1211)
                ii = 0
                endif
  130   WRITE(8,1210) DEN(1,I),DEN(2,I),den(3,i)
        goto 148
  135   write(8,1207)
        write(8,1208)
        ii = -1
        DO 138 I = 1,igen
        IF(DEN(1,I).gt. 1.) goTO 148
        ii = ii+1
```

```
         if(ii.eq. 3) then
                 write(8,1211)
                 ii = 0
                 endif
  138    WRITE(8,1212) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
  148    continue
C
 1200    FORMAT(1H ,5X,'Input:    ',6X,3x,'Mole fraction',4x,
         1       'CONCENTRATION OF C',6X,'GAS DENSITY')
 1205    FORMAT(1H ,14X,20x,2(13X,'kg/m**3'))
 1207    FORMAT(1H ,5X,'Adiabatic Mixing:',3x,'Mole fraction',3x,
         1       'CONCENTRATION OF C',6X,'GAS DENSITY',5x,
         1      6x,'Enthalpy',6x,4x,'Temperature')
 1208    FORMAT(1H ,14X,20x,2(13X,'kg/m**3'),7x,8x,'J/kg',8x,9x,'K')
 1210    FORMAT(1H ,14X,3(12X,F8.5))
 1211    format(1H )
 1212    FORMAT(1H ,14X,3(12X,F8.5),6x,3x,1pg13.5,7x,1pg13.5)
C
         WRITE(8,1111)
         write(8,1233) gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
      $ gas_ufl,gas_lfl,gas_zsp
         WRITE(8,1111)
         WRITE(8,1220) gmass0
         WRITE(8,1230)
C
         DO 150 I=1,IGEN
         IF(PTIME(I).EQ.POUNDN) GO TO 160
  150    WRITE(8,1231) PTIME(I),ET(I),R1T(I), PWC(I),PTEMP(I),
     $    PENTH(I)
  160    CONTINUE
C
 1220    FORMAT(1H ,'Source input data points',//,
     $           1h ,15x,'Initial mass in cloud: ',1pg13.5,//,
     $           1h ,7X,'Time',10X,'Contaminant',
     $           7X,'Source Radius',2x,3x,'Contaminant',4x,
     $           3x,'Temperature',4x,5x,'Enthalpy',5x,/,
     $           1x,18x,4x,'Mass Rate',5x,18x,2x,'Mass Fraction',3x)
 1230    FORMAT(1H ,9X,'s',15X,'kg/s',16X,'m',8x,' kg contam/kg mix ',8x,
     $           'K',9x,7x,'J/kg',7x)
 1231    FORMAT(1H ,6(3X,1PG12.5,3X))
 1233    Format(1H0,5x,'Specified Gas Properties:',//,
         110x,'Molecular weight:',T65,1pg13.5,/,
         110x,'Storage temperature:',T65,1pg13.5,'K',/,
         110x,'Density at storage temperature and ambient pressure:',
         1       T65,1pg13.5,'kg/m**3',/,
         110x,'Mean heat capacity constant:',T65,1pg13.5,/,
         110x,'Mean heat capacity power:',T65,1pg13.5,/,
         110x,'Upper mole fraction contour:',T65,1pg13.5,/,
         110x,'Lower mole fraction contour:',T65,1pg13.5,/,
         110x,'Height for isopleths:',T65,1pg13.5,'m',/)
 1241    format(1h0,5x,'Calculation procedure for ALPHA: ',I2)
 1242 format(1h0,5x,'Entrainment prescription for PHI: ',I2)
 1244 format(1h0,5x,'Layer thickness ratio used for average depth: ',
     1      1pg13.5)
 1250 format(1h0,5x,'Air entrainment coefficient used: ',f5.3)
 1251 format(1h0,5x,'NON Isothermal calculation')
 1252 format(1h0,5x,'Gravity slumping velocity coefficient used: ',f5.3)
 1253 format(1h0,5x,'Heat transfer calculated with fixed coefficient: ',
     1      1pg13.5,' J/m**2/s/K')
 1254 format(1h0,5x,'Heat transfer not included')
 1255 format(1h0,5x,'Heat transfer calculated with correlation: ',I2)
 1256 format(1h0,5x,'Isothermal calculation')
 1257 format(1h0,5x,'Water transfer calculated with fixed coefficient: ',
     1      1pg13.5,' /m**2/s/atm')
 1258 format(1h0,5x,'Water transfer not included')
 1259 format(1h0,5x,'Water transfer calculated with correlation')
C
      WRITE(8,1111)
      write(8,1241) ialpfl
      write(8,1242) iphifl
      write(8,1244) dellay
      write(8,1250) epsilon
      write(8,1252) ce
      if(isofl.eq. 0) write(8,1251)
      if(isofl.ne. 0) write(8,1256)
      if(ihtfl.lt. 0) write(8,1253) htco
      if(ihtfl.eq. 0) write(8,1254)
      if(ihtfl.gt. 0) write(8,1255) ihtfl
      if(iwtfl.lt. 0) write(8,1257) wtco
      if(iwtfl.eq. 0) write(8,1258)
      if(iwtfl.gt. 0) write(8,1259)
      WRITE(8,1111)
      iperm =1
      return
c
  190 continue
      if(.not.check4) return
      RAD = SQRT(2.*SLEN*SWID/pi)
      WRITE(8,1300) ESS,RAD
      WRITE(8,1320) SLEN,SWID
      qstar = ess/outl/outb/2.
      WRITE(8,1340) OUTCc,OUTSZ,qstar
      write(8,1350) swcl,swal,senl,srhl
      WRITE(8,1360) OUTL,OUTB
C
C
C
 1300 FORMAT(1H0,'Source strength [kg/s] : ',18X,1PG13.5,T60,
     $'Equivalent Primary source radius [m] : ',5x,1PG13.5)
 1320 FORMAT(1H ,'Equivalent Primary source length [m] : ',4X,1PG13.5,
     $T60,'Equivalent Primary source half-width [m] : ',1X,1PG13.5)
 1340 FORMAT(/,' Secondary source concentration [kg/m**3] : ',
     $1PG13.5,T60,'Secondary source SZ [m] : ',18X,1PG13.5,//,
        1 ' Contaminant flux rate: ',1pg13.5,/)
 1350   format(/,' Secondary source mass fractions... contaminant: ',
        1      1pg13.6,2x,' air: ',1pg13.5,/,' ',10x,' Enthalpy: ',
        1      1pg13.5,5x,' Density: ',1pg13.5,/)
 1360   FORMAT(1H ,'Secondary source length [m] : ',13X,1PG13.5,T60,
     $'Secondary source half-width [m] : ',10X,1PG13.5)
C
C
        RETURN
        END
####
```

```
c.........................................................
c
c      gamma and incomplete gamma function
c
c.........................................................
c
c      routines included in this file:
c             GAMINC        Incomplete gamma function
c             GAMMLN        Natural log of the gamma function
c             GSER          computation procedure
c             GCF           computation procedure
c
c.........................................................
c
c      These functions return the value for the incomplete Gamma function
c      with the arguments (ALPHA,BETA).  The functional values are
c      calculated with either a series representation or a continued
c      fraction representation.  These routines are based on:
c
c      Press, W.H., B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling,
c      "Numerical Recipes", Cambridge University Press, Cambridge,1986.
c
c
c
        function gaminc(alpha,beta)

        implicit Real*8 (a-h, o-z), integer*4 (i-n)
c
c
c*** ensure the arguments are within the proper bounds.
c
        if( beta.lt.0. .or. alpha.le.0.) then
               write(6,9000) alpha,beta
 9000          format(' GAMINC? Arugments are out-of-bounds. ALPHA: ',
     $ 1PG13.5,' BETA: ',1PG13.5)
               STOP
               endif
c
c*** determine which of the series or continued fraction representation
c    is more appropriate.
c
        if( beta .lt. alpha+1. ) then
c                                                       ! series
               call gser( aa, alpha, beta, gln)
        else
c                                                       ! continued fraction
               call gcf( aa, alpha, beta, gln)
               aa = 1. - aa
        endif
c
c*** multiply the result by GAMMA(ALPHA) to get the final value.
```

```
c
        gaminc = dexp( dlog(aa) + gln)
        return
        end
c
c
c
        subroutine gser( gamser, alpha, beta, gln)
c
c       This routine calculates the incomplete gamma function using
c       the series representation.
c
        implicit real*8 (a-h,o-z), integer*4 (i-n)
c
        parameter (itmax=100, eps=1.d-9)

        gln = gammln(alpha)

        if(beta .lt. 0.) then
                write(6,*) 'GSER? BETA is out-of-bounds.'
                stop
                endif
        if(beta .eq. 0.) then
                gamser = 0.
                return
                endif

        ap = alpha
        sum = 1./alpha
        del = sum

        do 100 n=1,itmax
                ap = ap + 1.
                del = del* beta /ap
                sum = sum + del
                if( dabs(del) .lt. dabs(sum)*eps) then
                        gamser = sum*dexp( -beta+alpha*dlog(beta)-gln)
                        return
                        endif
100     continue

        write(6,*) 'GSER? ALPHA is too large or ITMAX is too small.'
        stop
        end
c
c
c
        subroutine gcf( gamcf, alpha, beta, gln)
c
c       This routine calculates the incomplete gamma function using
c       the continued fraction representation.
```

```
c
        implicit real*8 (a-h,o-z), integer*4 (i-n)
c
        parameter (itmax=100, eps=1.d-9)

        gln = gammln(alpha)

c
c*** INITIALIZE some variables.
c
c                               ! previous value tested for convergence
        gold = 0.
        a0 = 1.
        a1 = beta
        b0 = 0.
        b1 = 1.
c
c                               ! fac is a "renormalization factor"
        fac = 1.

        do 100 n=1,itmax
                an = dble(n)
                ana = an - alpha
                a0 = (a1 + a0*ana)
                b0 = (b1 + b0*ana)
                anf = an
                a1 = beta *a0 + anf*a1
                b1 = beta *b0 + anf*b1
                if( a1 .ne. 0.) then
                        fac = 1./b1
                        gg = a1*fac
                a0 = a0*fac
                a1 = a1*fac
                b0 = b0*fac
                b1 = b1*fac
                        if( dabs((gg-gold)/gg) .lt .eps) then
                                gamcf = dexp(
     $                                  -beta+alpha*dlog(beta)-gln)/gg
                                return
                                endif
                        gold = gg
                endif
100     continue

        write(6,*) 'GCF? ALPHA is too large or ITMAX is too small.'
        stop
        end
c
c
c
        function gammln( alpha )
c
```

```
c       This routine calculates ln( gamma(alpha)) for alpha>0.
c
        implicit real*8 (a-h,o-z), integer*4 (i-n)
c
        dimension cof(6)

        data cof,stp/76.18009173D0, -86.50532033D0, 24.01409822D0,
     $ -1.231739516D0, 0.120858003D-2, -0.536382D-5, 2.50662827465D0/
        data half, one, fpf/0.5D0, 1.0D0, 5.5D0/
c
        if( alpha .lt. 1) then
                gam = gamma( alpha )
                gammln = dlog(gam)
                return
                endif
c
        xx = alpha - one
        tmp = xx + fpf
        tmp = (xx + half) * dlog(tmp) - tmp
        ser = one
        do 100 j=1,6
                xx = xx + one
                ser = ser + cof(j)/xx
100     continue

        gammln = tmp + dlog(stp*ser)
        return
        end
####
```

```
C.............................................................
C
C       INPUT SUBROUTINE FOR DEGADIS MODEL
C
        SUBROUTINE IO(tend,gmass0,OPNRUP)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
c
        include 'sys$degadis:DEGADIS1.dec'
c
C       BLOCK COMMON
C
        COMMON
     $/TITL/TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/PARM/ U0,Z0,ZR,rML,USTAR,vkc,gg,RHOE,RHOA,DELTAy,BETAy,GAMMAF,
     .          CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl
     $/phlag/check1,check2,again,check3,check4,check5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/NEND/ POUNDN,POUND
     $/ICNTL/ IPRINT,IOBS(10)
     ./oomsin/ oodist,avtime
C
C
        character*80 TITLE(4)
        character*4 pound
        character*24 TSRC,TINP,TOBS,TSRT
        character*3 gas_name
C
        logical check1,check2,again,check3,check4,check5
C
        character*40 OPNRUP
C
        OPEN(UNIT=9, NAME=OPNRUP, TYPE='OLD')
C
        DO 90 I=1,4
        READ(9,2000) TITLE(I)
   90   CONTINUE
 2000   FORMAT(A80)
C
        READ(9,*) U0,Z0,zr
        read(9,*) istab
        read(9,*) oodist, avtime
        READ(9,*) DELTAy,BETAy,rml
        read(9,*) sigx_coeff,sigx_pow,sigx_min_dist
        read(9,*) tamb,pamb,humid
        humsrc = 0.
        read(9,*) isofl,tsurf
        read(9,*) ihtfl,htco
        read(9,*) iwtfl,wtco
        read(9,2020) gas_name
 2020   FORMAT(A3)
        read(9,*) gas_mw,gas_temp,gas_rhoe
        read(9,*) gas_cpk,gas_cpp
        read(9,*) gas_ufl,gas_lfl,gas_zsp
c
        if(isofl .eq. 0) then
              rhoe = gas_rhoe
          rhoa = pamb*(1.D0+humid)/(.002833D0+ 0.004553D0*humid)/tamb
              goto 105
              endif
        READ(9,*) NP
        DO 100 I=1,NP
 100    READ(9,*) DEN(1,I),DEN(2,I),den(3,I),den(4,i),den(5,i)
        RHOE = DEN(3,NP)
        RHOA = DEN(3,1)
        den(1,np+1) = 2.
c
 105    READ(9,*) CcLOW
c
        read(9,*) gmass0
        READ(9,*) NP
        DO 110 I=1,NP
        READ(9,*) PTIME(I),ET(I),R1T(I), PWC(I),PTEMP(I),PFRACV(I)
        WA = (1.D0 - PWC(I))/(1.D0 + HUMID)
        PENTH(I) = ENTHAL( PWC(I), WA, PTEMP(I))
        CALL TPROP(-1,PWC(I), WA,PENTH(I), YC,YA,WM, PTEMP(I), PRHO(I),CP)
 110    CONTINUE
        TEND = PTIME(NP-2)
        PTIME(NP + 1)  = POUNDN
c
        read(9,*) check1,check2,again,check3,check4,check5
c
        tobs = ' '
        tsrt = ' '
        READ(9,2010) TINP
 2010   format(a24)
c
        if(check4) read(9,*) ess,slen,swid
c
        CLOSE(UNIT=9)
        RETURN
        END
####
```

```
      SUBROUTINE IOT(OPNRUP)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      include 'sys$degadis:DEGADISIN.dec'
c
      COMMON
     $/TITL/ TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/PARM/ U0,Z0,ZR,rML,USTAR,vkc,gg,RHOE,RHOA,DELTAy,BETAy,GAMMAF,
     .          CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/NEND/ POUNDN,POUND
     ./oomsin/ oodist,avtime
C
      character*80 TITLE(4)
      character*3 gas_name
      character*4 pound
      character*24 TSRC,TINP,TOBS,TSRT
C
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      character*(*) OPNRUP
      character*40 STRING
      character*4 dummy
C
      WRITE(6,1100)
      WRITE(6,1110)
C
C*** OPEN THE INPUT FILE
C
      OPEN(UNIT=8,NAME=OPNRUP,TYPE='NEW',
     $ carriagecontrol='list',recordtype='variable')
C
C*** NOW GET THE TITLE BLOCK
C
      WRITE(6,1120)
      WRITE(6,1130)
C
      DO 100 I=1,4
      READ(5,1134) TITLE(I)
      dummy = title(I)
      IF(dummy(1:4) .EQ. POUND(1:4)) GO TO 110
```

```
        WRITE(8,1135) TITLE(I)
  100   CONTINUE
        GO TO 130
 C
  110   CONTINUE                         ! FILL OUT THE BLOCK
        II = I
        DO 120 I = II,4
        TITLE(I) = ' '
        WRITE(8,1135) TITLE(I)
  120   CONTINUE
  130   CONTINUE
 C
 c*** Atmospheric parameters:
 c
        WRITE(6,1140)
        WRITE(6,1142)
        READ(5,*) U0,Z0,ZR
        WRITE(8,1020) U0,Z0,ZR
 C
 c*** stability, averaging time for DELTAY, and derived parameters
 c
        WRITE(6,1150)
        READ(5,1310) NCHAR,STRING
        write(6,1152)
        read(5,*) avtime
        istab = 4                                ! default is D stability
        IF(STRING.eq.'A' .or. string.eq.'a') istab=1
        IF(STRING.eq.'B' .or. string.eq.'b') istab=2
        IF(STRING.eq.'C' .or. string.eq.'c') istab=3
        IF(STRING.eq.'D' .or. string.eq.'d') istab=4
        IF(STRING.eq.'E' .or. string.eq.'e') istab=5
        IF(STRING.eq.'F' .or. string.eq.'f') istab=6
        goto(161,162,163,164,165,166) istab
 161    timeav = dmax1( avtime, 18.4D0)                  ! A
        deltay = 0.423D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        rml = -11.43D0 * zr**0.103D0
        sigx_coeff = 0.02
        sigx_pow = 1.22
        sigx_min_dist = 130.
        goto 170
 162    timeav = dmax1( avtime, 18.4D0)                  ! B
        deltay = 0.313D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        rml = -25.98D0 * zr**0.171D0
        sigx_coeff = 0.02
        sigx_pow = 1.22
        sigx_min_dist = 130.
        goto 170
 163    timeav = dmax1( avtime, 18.4D0)                  ! C
        deltay = 0.210D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        rml = -123.4D0 * zr**0.304D0
        sigx_coeff = 0.02
        sigx_pow = 1.22
        sigx_min_dist = 130.
        goto 170
164     timeav = dmax1( avtime, 18.3D0)                 ! D
        deltay = 0.1360D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        rml = 0.0               ! used for infinity
        sigx_coeff = 0.04
        sigx_pow = 1.14
        sigx_min_dist = 100.
        goto 170
165     timeav = dmax1( avtime, 11.4D0)                 ! E
        deltay = 0.1020D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        rml = 123.4D0 * zr**0.304D0
        sigx_coeff = 0.17
        sigx_pow = 0.97
        sigx_min_dist = 50.
        goto 170
166     timeav = dmax1( avtime, 4.6D0)                  ! F
        deltay = 0.0674D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        rml = 25.98D0 * zr**0.171D0
        sigx_coeff = 0.17
        sigx_pow = 0.97
        sigx_min_dist = 50.
C
170     WRITE(8,1040) istab
        oodist = 0.D0
        write(8,1020) oodist,avtime
C
172     if( rml.ne. 0.) then
        WRITE(6,1159) deltay,betay,rml,sigx_coeff,sigx_pow,sigx_min_dist
        else
        WRITE(6,1160) deltay,betay,sigx_coeff,sigx_pow,sigx_min_dist
        endif
        read(5,1310) nchar,string
        if(string.eq.'d' .or. string.eq.'D') then
                write(6,1600)
                read(5,*) deltay
                goto 172
        else if(string.eq.'b' .or. string.eq.'B') then
                write(6,1620)
                read(5,*) betay
                goto 172
        else if(string.eq.'l' .or. string.eq.'L') then
                write(6,1660)
                read(5,*) rml
                goto 172
        else if(string.eq.'c' .or. string.eq.'C') then
                write(6,1670)
                read(5,*) sigx_coeff
                goto 172
        else if(string.eq.'p' .or. string.eq.'P') then
                write(6,1680)
                read(5,*) sigx_pow
                goto 172
        else if(string.eq.'m' .or. string.eq.'M') then
                write(6,1690)
                read(5,*) sigx_min_dist
                goto 172
        else if(nchar.eq.0 .or. string.eq.'n' .or. string.eq.'N') then
                WRITE(8,1020) DELTAY,BETAY,rml
                WRITE(8,1020) sigx_coeff,sigx_pow,sigx_min_dist
        else
        goto 172
        endif
c
c*** ambient pressure, temperatures, and humidity
c
        write(6,1500)
        read(5,*) tamb,pamb
        vaporp = 6.0298e-3* exp(5407.* (1./273.15- 1./tamb)) ! atm
        sat = 0.622*vaporp / (pamb- vaporp)
        write(6,1580)
        read(5,1310) nchar,string
        if(string.eq.'a' .or. string.eq.'A') then
                write(6,1585)
                read(5,*) humid
                relhumid = 100.*humid/sat
                write(6,1586) relhumid
                goto 200
                endif
        write(6,1587)
        read(5,*) relhumid
        humid = relhumid/100. * sat
 200    rhoa = pamb*(1.+humid)/(.002833+.004553*humid)/tamb
        write(6,1588) rhoa
        write(8,1025) tamb,pamb,humid, relhumid
c
        isofl = 0
        ihtfl = 0
        htco  = 0.
        iwtfl = 0
        wtco  = 0.
        tsurf = tamb
c
        write(6,2000)
        read(5,1310) nchar,string
        if(string.eq.'Y' .or. string.eq.'y') then
                isofl = 1
                tsurf = tamb
                goto 250
                endif
c
        write(6,2020)
        read(5,1310) nchar,string
        if(string.eq.'Y' .or. string.eq.'y') then
                write(6,2030)
                read(5,*) tsurf
 220            write(6,2040)
                read(5,1310) nchar,string
                        if(string.eq.'V' .or. string.eq.'v') then
                        ihtfl = -1      ! constant value
                        write(6,2050)
                        read(5,*) htco
        else if(string.eq.'C' .or. string.eq.'c' .or. nchar.eq.0) then
                                ihtfl = 1       ! local correlation
                        else if(string.eq.'L' .or. string.eq.'l') then
                                ihtfl = 2       ! LLNL correlation
                                htco = 0.0125   ! [=]m/s
                                write(6,2043) htco
                                read(5,1310) nchar,string
                                if(string.eq.'Y' .or. string.eq.'y')
        1
                                        read(5,*) htco
                        else
                        goto 220
                        endif
        else
        goto 250
        endif
c
        write(6,2100)
        read(5,1310) nchar,string
        if(string.eq.'Y' .or. string.eq.'y') then
                iwtfl = 1
                write(6,2045)
                read(5,1310) nchar,string
                        if(string.eq.'V' .or. string.eq.'v') then
                        iwtfl = -1
                        write(6,2120)
                        read(5,*) wtco
                        endif
                endif
c
 250    continue
        write(8,1060) isofl,tsurf
        write(8,1060) ihtfl,htco
        write(8,1060) iwtfl,wtco
c
```

```
c
c*** gas characteristics
c
        write(6,1510)
        read(5,1415) gas_name
        write(8,1415) gas_name
        gas_mw = 16.04
        gas_temp = 111.7
        gas_rhoe = 1.792*pamb   ! correct to pamb
        gas_cpk = 2730.
        gas_cpp = 1.00
        gas_ufl= 0.15
        gas_lfl= 0.05
        gas_zsp= 0.5

        if(gas_name.eq.'NH3' .or. gas_name.eq.'nh3') then
                gas_mw = 17.
                gas_temp = tamb
                gas_rhoe = pamb*gas_mw/0.08205/tamb     ! ideal gas
                gas_cpk = 3845.         ! to get 807.5 J/kg/K
                gas_cpp = 1.000
                gas_ufl= 2.0E-2
                gas_lfl= 2.00E-3
                gas_zsp= 0.5

        else if(gas_name.eq.'LNG' .or. gas_name.eq.'lng') then
                gas_mw = 16.04
                gas_temp = 111.7
                gas_rhoe = 1.792*pamb   ! correct to pamb
                gas_cpk = 5.6e-8
                gas_cpp = 5.00
                gas_ufl= 0.15
                gas_lfl= 0.05
                gas_zsp= 0.5

        else if(gas_name.eq.'LPG' .or. gas_name.eq.'lpg') then
                gas_mw = 44.09
                gas_temp = 231.
                gas_rhoe = 2.400*pamb   ! correct to pamb
                gas_cpk = 15.4
                gas_cpp = 2.25
                gas_ufl= 0.10
                gas_lfl= 0.02
                gas_zsp= 0.5

        else if(gas_name.eq.'NO2' .or. gas_name.eq.'no2') then
                gas_mw = 46.
                gas_temp = tamb
                gas_rhoe = pamb*gas_mw/0.08205/tamb     ! ideal gas
                gas_cpk = 3845.         ! to get 807.5 J/kg/K
                gas_cpp = 1.000
                gas_ufl= 1000.0E-6
                gas_lfl= 500.00E-6
                gas_zsp= 0.5

        else if(gas_name.eq.'N2O' .or. gas_name.eq.'n2o') then
                gas_mw = 92.
                gas_temp = tamb
                gas_rhoe = pamb*gas_mw/0.08205/tamb     ! ideal gas
                gas_cpk = 40990.        ! to get 807.5 J/kg/K
                gas_cpp = 1.000
                gas_ufl= 1000.0E-6
                gas_lfl= 500.00E-6
                gas_zsp= 0.5

        else if(gas_name.eq.'CL2' .or.
     1          gas_name.eq.'cl2' .or. gas_name.eq.'Cl2') then
                gas_mw = 70.91
                gas_temp = 238.7
                gas_rhoe = 3.672*pamb   ! correct to pamb
                gas_cpk = 484.2
                gas_cpp = 1.000
                gas_ufl= 0.1D-4
                gas_lfl= 0.3D-5
                gas_zsp= 0.5

        else if(gas_name.eq.'MEC' .or.gas_name.eq.'mec') then
                gas_mw = 84.94
                gas_temp = tamb
                gas_rhoe = 3.53*(298.15/tamb)*pamb ! correct to pamb,tamb
                gas_cpk = 2730.         ! Open to question?
                gas_cpp = 1.000
                gas_ufl= 1000.D-6
                gas_lfl= 500.D-6
                gas_zsp= 0.5
        endif
 270    write(6,1520) gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     1          gas_ufl,gas_lfl,gas_zsp
        read(5,1310) nchar,string
        if(string.eq.'m' .or. string.eq.'M') then
                write(6,1550)
                read(5,*) gas_mw
                goto 270
        else if(string.eq.'t' .or. string.eq.'T') then
                write(6,1530)
                read(5,*) gas_temp
                goto 270
        else if(string.eq.'d' .or. string.eq.'D') then
                write(6,1535)
                read(5,*) gas_rhoe
                goto 270
        else if(string.eq.'h' .or. string.eq.'H') then
```

```
                write(6,1570)
                read(5,*) gas_cpk
                goto 270
          else if(string.eq.'p' .or. string.eq.'P') then
                write(6,1571)
                read(5,*) gas_cpp
                goto 270
          else if(string.eq.'u' .or. string.eq.'U') then
                write(6,1572)
                read(5,*) gas_ufl
                goto 270
          else if(string.eq.'l' .or. string.eq.'L') then
                write(6,1573)
                read(5,*) gas_lfl
                goto 270
          else if(string.eq.'z' .or. string.eq.'Z') then
                write(6,1574)
                read(5,*) gas_zsp
                goto 270
          else if(nchar.eq. 0 .or. string.eq.'n' .or. string.eq.'N') then
             if(gas_cpp .eq. 0.D0) then
                gas_cpp = 1.D0
                gas_cpk = gas_cpk*gas_mw - 3.3304
             endif
                WRITE(8,1020) gas_mw, gas_temp,gas_rhoe
                write(8,1020) gas_cpk,gas_cpp
                WRITE(8,1020) gas_ufl,gas_lfl,gas_zsp
          else
          goto 270
          endif
c
c density curve if isothermal
c
          if(isofl .eq. 0) goto 460
          WRITE(6,1161)
          WRITE(6,1162)
          WRITE(6,1163)
          WRITE(6,1164) rhoe
          WRITE(6,1165)
          WRITE(6,1166)
          goto 320
C
 280      write(6,1290)
C
 320      LUNIN = 5
          WRITE(6,1300)
          READ(5,1310) NCHAR,STRING
          IF(STRING.EQ.'y' .or. string.eq.'Y') GO TO 360
          GO TO 400
 360      WRITE(6,1320)
          READ(5,1310) NCHAR,STRING
```

```
      OPEN(UNIT=10,NAME=STRING,TYPE='OLD',err=280)
      LUNIN = 10
 400  CONTINUE
      IF(LUNIN .EQ. 5) WRITE(6,1170) igen
      READ(LUNIN,*) NP
      WRITE(8,1040) NP
      IF(LUNIN .EQ. 5) WRITE(6,1180)
C
      DO 440 I=1,NP
      den(4,i) = 0.                             ! 0.0 by default for isotherm
      den(5,i) = tamb                           ! tamb for isotherm
      READ(LUNIN,*) DEN(1,I),DEN(2,I),DEN(3,I)
      if(i .eq.1 .and.
     1       (den(3,1)/rhoa.gt.1.005 .or. rhoa/den(3,1).gt.1.005)) then
             den(3,1) = rhoa
             write(6,1340) rhoa
             endif
      if(i.eq.np) THEN
             IF(     den(2,i)/gas_rhoe .gt. 1.005
     1       .or.    gas_rhoe/den(2,i) .gt. 1.005
     1       .or.    den(3,i)/gas_rhoe .gt. 1.005
     1       .or.    gas_rhoe/den(3,i) .gt. 1.005) then
             den(2,i) = gas_rhoe
             den(3,i) = gas_rhoe
             write(6,1341) gas_rhoe
             endif
             endif
      WRITE(8,1025) DEN(1,I),DEN(2,I),DEN(3,I),Den(4,I),den(5,i)
 440  CONTINUE
      IF(LUNIN .EQ. 10) CLOSE(UNIT=10)
C
c
 460  ccmax = (gas_lfl/2.) * pamb*gas_mw/0.08205/tamb
      WRITE(6,1280) ccmax
      READ(5,*) CcLOW
      if(cclow .le. 0.) cclow=0.005   ! don't let 0. get through
      if(cclow .gt. ccmax) cclow=ccmax
      WRITE(8,1010) CcLOW
c
c
c*** source description
c
      write(6,1440)
c
      write(6,1460)
      read(5,1410) dummy
      if(dummy.eq.'d' .or. dummy.eq.'D') goto 730
c
      check4 = .false.
      write(6,1400)
      read(5,1410) dummy
```

```
         if(dummy.eq.'y' .or. dummy.eq.'Y') goto 480
         goto 520
  480    continue
         gmass0 = 0.              ! no initial cloud for a SS simulation
         write(8,1020) gmass0
         write(6,1420)
         read(5,*) ess
         write(6,1430)
         read(5,*) r1ss
         np = 4
c
         tend = 60230. ! [=] sec
c
         PTIME(1) = 0.
         et(1) = ess
         r1t(1)= r1ss
         pwc(1) = 1.
         ptemp(1) = gas_temp
         pfracv(1) = 1.
         PTIME(2) = tend
         et(2) = ess
         r1t(2)= r1ss
         pwc(2) = 1.
         ptemp(2) = gas_temp
         pfracv(2) = 1.
         PTIME(3) = tend + 1.
         et(3) = 0.
         r1t(3)= 0.
         pwc(3) = 1.
         ptemp(3) = gas_temp
         pfracv(3) = 1.
         PTIME(4) = tend + 2.
         et(4) = 0.
         r1t(4)= 0.
         pwc(4) = 1.
         ptemp(4) = gas_temp
         pfracv(4) = 1.
         slen = 2.*r1ss
         swid = pi*r1ss**2/slen/2.
         check4 = .true.        ! steady state run
         goto 790
c
  520    continue
c
         write(6,1450)
         read(5,*) gmass0
         write(8,1020) gmass0
C
         WRITE(6,1190)
         WRITE(6,1200)
         WRITE(6,1210)
```

```
        WRITE(6,1220)
        WRITE(6,1221)
        WRITE(6,1165)
        WRITE(6,1223)
        WRITE(6,1230)
        WRITE(6,1240)
        WRITE(6,1250)
        goto 600
C
 560    write(6,1290)
C
 600    LUNIN = 5
        WRITE(6,1330)
        READ(5,1310) NCHAR,STRING
        IF(STRING.eq.'Y' .or. string.eq.'y') goto 640
        goto 680
  640   WRITE(6,1320)
        READ(5,1310) NCHAR,STRING
        OPEN(UNIT=10,NAME=STRING,TYPE='OLD',err=560)
        LUNIN = 10
 680    CONTINUE
        IF(LUNIN .EQ. 5) WRITE(6,1260) igen
        READ(LUNIN,*) NP
        IF(LUNIN .EQ. 5) WRITE(6,1270)
C
        DO 720 I=1,NP
        READ(LUNIN,*) PTIME(I),ET(I),R1T(I)
        pwc(i) = 1.
        ptemp(i) = gas_temp
        pfracv(i) = 1.
 720    CONTINUE
        IF(LUNIN .EQ. 10) CLOSE(UNIT=10)
        GOTO 790
C
C*** FOR A DILUTED SOURCE ...
C
 730    check4 = .false.
        write(6,1400)
        read(5,1410) dummy
        if(dummy.eq.'y' .or. dummy.eq.'Y') goto 740
        goto 750
 740    continue
        gmass0 = 0.             ! no initial cloud for a SS simulation
        write(8,1020) gmass0
        write(6,1420)
        read(5,*) ess
        write(6,1430)
        read(5,*) r1ss
 741    WRITE(6,1470)
        read(5,*) PWC(1)
        if(pwc(1).le.0.D0 .or. pwc(1).gt.1.D0) goto 741
 742    WRITE(6,1480)
        read(5,*) PTEMP(1)
        if(pwc(1).le.0.D0) goto 742
        np = 4
c
        tend = 60230. ! [=] sec
c
        PTIME(1) = 0.
        et(1) = ess
        r1t(1)= r1ss
        PWC(1)= pwc(1)
        PTEMP(1)= ptemp(1)
        PFRACV(1)= 1.0
        PTIME(2) = tend
        et(2) = ess
        r1t(2)= r1ss
        PWC(2)= pwc(1)
        PTEMP(2)=ptemp(1)
        PFRACV(2)= 1.0
        PTIME(3) = tend + 1.
        et(3) = 0.
        r1t(3)= 0.
        PWC(3)= pwc(1)
        PTEMP(3)=ptemp(1)
        PFRACV(3)= 1.0
        PTIME(4) = tend + 2.
        et(4) = 0.
        r1t(4)= 0.
        PWC(4)= pwc(1)
        PTEMP(4)=ptemp(1)
        PFRACV(4)= 1.0
        slen = 2.*r1ss
        swid = pi*r1ss**2/slen/2.
        check4 = .true.         ! steady state run
        goto 790
c
 750    continue
c
        write(6,1450)
        read(5,*) gmass0
        write(8,1020) gmass0
C
        WRITE(6,1190)
        WRITE(6,2200)
        WRITE(6,2210)
        WRITE(6,2220)
        WRITE(6,2221)
        WRITE(6,1165)
        WRITE(6,2223)
        WRITE(6,2230)
        WRITE(6,2240)
```

```
      WRITE(6,2250)
      goto 760
C
  755 write(6,1290)
C
  760 LUNIN = 5
      WRITE(6,1330)
      READ(5,1310) NCHAR,STRING
      IF(STRING.eq.'Y' .or. string.eq.'y') goto 765
      goto 770
  765 WRITE(6,1320)
      READ(5,1310) NCHAR,STRING
      OPEN(UNIT=10,NAME=STRING,TYPE='OLD',err=755)
      LUNIN = 10
  770 CONTINUE
      IF(LUNIN .EQ. 5) WRITE(6,1260) igen
      READ(LUNIN,*) NP
      IF(LUNIN .EQ. 5) WRITE(6,1270)
C
      DO 780 I=1,NP
      PFRACV(I) = 1.
      READ(LUNIN,*) PTIME(I),ET(I),R1T(I), PWC(I),PTEMP(I)
  780 CONTINUE
      IF(LUNIN .EQ. 10) CLOSE(UNIT=10)
C
  790 continue
      WRITE(8,1040) NP
      DO 800 I=1,NP
  800 WRITE(8,1030) PTIME(I),ET(I),R1T(I), PWC(I),PTEMP(I),PFRACV(I)
C
      if(et(1).eq.0. .and. gmass0.ne.0.) check2=.true. ! HSE type spill
      write(8,*) check1,check2,again,check3,check4,check5
C
      istat = lib$date_time(tinp)
      WRITE(8,1050) TINP
c
      if(check4) write(8,1020) ess,slen,swid          ! steady state
c
C
      CLOSE(UNIT=8)
C
c
 1010 format(1x,1pg14.7)
 1020 format(3(1x,1pg14.7))
 1025 format(5(1x,1pg14.7))
 1030 format(1x,5(1pg14.7,1x),1pg14.7)
 1040 format(1x,I5)
 1050 format(a24)
 1060 format(1x,i4,1x,1pg14.7)
c
 1100 FORMAT(5X,'INPUT MODULE -- DEGADIS MODEL')
```

```
1110   FORMAT(/,5X,'*******************************************')
1120   FORMAT(5X,'Enter Title Block -- up to 4 lines of 80',
     $' characters')
1130   FORMAT(5X,'To stop, type "//"')
1134   FORMAT(A80)
1135   FORMAT(A80)
1140   FORMAT(5X,'ENTER WIND PARAMETERS -- U0 (m/s), Z0 (m), ',
     $'and ZR(m)')
1142   format(5x,'U0 -- Wind velocity at reference height Z0',
     $/,5X,'ZR -- Surface Roughness')
1150   FORMAT(/,5X,'Enter the Pasquill stability class: (A,B,C,',
     $'D,E,F) <D> ',$)
1152   format(' Enter the averaging time (s) for estimating DELTAY: ',$)
1159   format(/,' The values for the atmospheric parameters',
     $' are set as follows:',
     $/,' DELTAY:                    ',F12.4,
     $/,' BETAY:                     ',F12.4,
     $/,' Monin-Obukhov length:      ',F12.4,' m',
     $/,' Sigma X Coefficient:       ',F12.4,
     $/,' Sigma X Power:             ',F12.4,
     $/,' Sigma X Minimum Distance: ',F12.4,' m',
     $/,' Do you wish to change any of these?',
     $/,' (No,Deltay,Betay,Length,Coefficient,Power,Minimum) <N> ',$)
1160   format(/,' The values for the atmospheric parameters',
     $' are set as follows:',
     $/,' DELTAY:                    ',F12.4,
     $/,' BETAY:                     ',F12.4,
     $/,' Monin-Obukhov length:      infinite',
     $/,' Sigma X Coefficient:       ',F12.4,
     $/,' Sigma X Power:             ',F12.4,
     $/,' Sigma X Minimum Distance: ',F12.4,' m',
     $/,' Do you wish to change any of these?',
     $/,' (No,Deltay,Betay,Length,Coefficient,Power,Minimum) <N> ',$)
1161   FORMAT(/,5X,'The density is determined as a function of con',
     $'centration')
1162   FORMAT(5X,'by a listing of ordered triples supplied by the ',
     $'user.')
1163   FORMAT(5X,'Use the following form:')
1164   FORMAT(/,5X,5X,'first point',6X,'-- pure air  y=0.0,Cc=0.,',
     1      'RHOG=RHOA=',F7.5' kg/m**3')
1165   FORMAT(3(15X,'.',/))
1166   FORMAT(5X,5X,'last point',7x,'-- pure gas  y=1.0,Cc=RHOE,',
     1      'RHOG=RHOE')
1170   FORMAT(/,5X,'ENTER THE NUMBER OF DATA TRIPLES (max=',i2,')',
     $' FOR THE DENSITY FUNCTION: ',$)
1180   FORMAT(/,5X,'Enter Mole frac, Cc (kg/m**3), then RHOG ',
     1      '(kg/m**3) by triples')
c
c
1190   FORMAT(/,5X,10X,'Source Description')
1200   FORMAT(1X,/,5X,'The description of the primary source mass')
```

```
 1210   FORMAT(5X,'evolution rate E and radius R1 for a transient',/,
     $5x,'release is input by ordered triples as follows:')
 1220   FORMAT(/,5X,3X,'first point',8X,
     $'-- t=0, E(t=0), R1(t=0) (initial, ',
     $'nonzero values)')
 1221   FORMAT(5X,3X,'second point',7X,
     $'-- t=t1, E(t=t1), R1(t=t1)')
 1223   FORMAT(5X,3X,'last nonzero point -- ',
     $'t=TEND, E(t=TEND), R1(t=TEND)')
 1230   FORMAT(5X,3X,'next to last point -- t=TEND+1., E=0., R1=0.')
 1240   FORMAT(5X,3X,'last point         -- t=TEND+2., E=0., R1=0.')
 1250   FORMAT(/,5X,'Note: the final time (TEND) is the last time ',
     $'when E and R1 are non-zero.',/)
 1260   FORMAT(/,5X,'Enter the number of triples (max= ',i2,')',
     $' starting with t=0. and ending',/,5x,'with t=TEND+2.',
     $' for the source description: ',$)
 1270   FORMAT(/,5X,'Enter TIME (sec), EVOLUTION RATE (kg/s), ',
     $'and POOL RADIUS (m)')
 1280   FORMAT(/,5X,'The suggested LOWEST CONCENTRATION OF INTEREST ',
     $'(gas_lfl/2.)',/,5x,' is ',1pg13.5,
     $' kg/m**3.  Enter the desired value: ',$)
 1290   format(/,' This file was not found.')
 1300   FORMAT(/,' Do you have an input file for the Density ',
     $'function? [y or N] ',$)
 1310   FORMAT(Q,A20)
 1320   FORMAT(' Enter the file name: [DIR]FILE_NAME.EXT ',$)
 1330   FORMAT(' Do you have an input file for the Source ',
     $'Description? [y or N] ',$)
 1340   format(/,' Air density corrected to ',1pg13.5' kg/m**3',/)
 1341   format(/,' Contaminant density corrected to ',1pg13.5' kg/m**3',/)
c
c
 1400   format(//,' Is this a Steady state simulation? <y or N> ',$)
 1410   format(a4)
 1415   format(a3)
 1420   format(/,' Enter the desired evolution rate [=] kg/sec : ',$)
 1430   format(' Enter the desired source radius [=] m : ',$)
 1440   format(/,' Specification of source parameters.',/)
 1450   format(//,' Enter the initial mass of pure gas',
     $' over the source. (kg)',/,' (Positive or zero): ',$)
 1460   format(/,' Is this a release of pure (P) or diluted (d) material',
     $          ' specified above? <P or d> ',$)
 1470   FORMAT(' Enter the desired primary source contaminant mass ',
     $          'fraction: ',$)
 1480   FORMAT(' Enter the desired primary source temperature [=] K : ')
c
c
 1500   format(/,' Enter the ambient temperature(K) and pressure',
     1          '(atm): ',$)
 1510   format(/,' Enter the code name of the diffusing species: ',$)
 1520   format(/,' The characteristics for the gas are set as follows:',/,
     $' Molecular weight:                                  ',f7.2,/,
     $' Storage temperature [K]:                           ',1pg13.5,/,
     $' Density at storage temperature, PAMB  [kg/m**3]: ',1pg13.5,/,
     $' Mean Heat capacity constant                        ',1pg13.5,/,
     $' Mean Heat capacity power                           ',1pg13.5,/,
     $' Upper Flammability Limit [mole frac]               ',1pg13.5,/,
     $' Lower Flammability Limit [mole frac]               ',1pg13.5,/,
     $' Height of Flammability Limit [m]                   ',1pg13.5,/,
     $' Do you wish to change any of these? ',
     $'(No,Mole,Temp,Den,Heat,Power,Upper,Lower,Z)',
     $' <N> ',$)
 1530   format(' Enter the desired Storage Temperature: ',$)
 1535   format(' Enter the desired Density at Storage ',
     1          'Temperature and',' ambient pressure: ',$)
 1550   format(' Enter the desired Molecular Weight: ',$)
 1570   format(' Enter the desired Mean Heat Capacity constant: ',$)
 1571   format(' Enter the desired Mean Heat Capacity power: ',$)
 1572   format(' Enter the desired Upper Flammability Limit: ',$)
 1573   format(' Enter the desired Lower Flammability Limit: ',$)
 1574   format(' Enter the desired Height for the flammable limit calcula',
     1          'tions: ',$)
 1580   format(/,' The ambient humidity can be entered as Relative ',
     1          'or Absolute.',/,' Enter either R or A <R or a>: ',$)
 1585   format(' Enter the absolute humidity (kg water/kg BDA): ',$)
 1586   format(' This is a relative humidity of ',1pg13.5,' %')
 1587   format(' Enter the relative humidity (%): ',$)
 1588   format(/,' Ambient Air density is ',1pg15.7,' kg/m**3')
c
c
 1600   format(' Enter the desired DELTAY: ',$)
 1620   format(' Enter the desired BETAY: ',$)
 1660   format(' Note: For infinity, RML = 0.0',/,
     $  ' Enter the desired Monin-Obukhov length: (m) ',$)
 1670   format(' Enter the desired Sigma X Coefficient: ',$)
 1680   format(' Enter the desired Sigma X Power: ',$)
 1690   format(' Enter the desired Sigma X Minimum distance: (m) ',$)
c
 2000   format(/,' Is this an Isothermal spill? <y or N> ',$)
 2020   format(/,' Is heat transfer to be included in the',
     1          ' calculations <y or N> ',$)
 2030   format(' Enter the surface temperature [=] K : ',$)
 2040   format(' Do you want to use the built in correlation,',
     1          ' the LLNL correlation, or',/,' enter',
     1          ' a particular value?',/,
     1          ' (Corr,LLNLcorr,Value)  <C> ',$)
 2043   format(/,' The form of the correlation is:',/,
     1          6x,'Q = (Vh * rho * cp) * area * (tsurf-temp)',//,
     1          ' with Vh = ',1pg13.5,' m/s.',//,
     1          ' Do you wish to change the value of Vh? (y or N): ',$)
 2045   format(' Do you want to use the built in correlation or enter',
     1          ' a particular value?',/,' <C or v> ',$)
```

```
2050   format(' Enter the HT coefficient value [=] J/m**2/s/K : ',$)
2100   format(/,' Is water transfer to be included in the',
      1       ' source <y or N> ',$)
2120   format(' Enter the WT coefficient value [=] kg/m**2/s : ',$)
C
2200   FORMAT(1X,/,5X,'The description of the primary source contam-')
2210   FORMAT(5X,'inant mass rate E, radius R1, contaminant mass',/,
     $5x,'fraction Wc,s, and temperature Ts for a transient',/,
     $5x,'release is input as a function of time as follows:')
2220   FORMAT(/,5X,3X,'first point',8X,
     $'-- t=0., E(t=0), R1(t=0), Wc,s(t=0), Ts(t=0)',/,
     $         5x,3x,18x,10x,'(initial, ',
     $'nonzero values)')
2221   FORMAT(5X,3X,'second point',7X,
     $'-- t=t1, E(t=t1), R1(t=t1), Wc,s(t=t1), Ts(t=t1)')
2223   FORMAT(5X,3X,'last nonzero point -- ',
     $'t=TEND, E(t=TEND), R1(t=TEND), Wc,s(t=TEND), Ts(t=TEND)')
2230   FORMAT(5X,3X,'next to last point -- t=TEND+1., E=0., R1=0.,',
     $         ' Wc,s=1., Ts=Tamb')
2240   FORMAT(5X,3X,'last point         -- t=TEND+2., E=0., R1=0.,',
     $         ' Wc,s=1., Ts=Tamb')
2250   FORMAT(/,5X,'Note: the final time (TEND) is the last time ',
     $'when E and R1 are non-zero.',/)
2260   FORMAT(/,5X,'Enter the number of times (max= ',i2,')',
     $' starting with t=0. and ending',/,5x,'with t=TEND+2.',
     $' for the source description: ',$)
2270   FORMAT(/,5X,'Enter TIME (sec), CONTAMINANT MASS RATE (kg C/s), ',
     $'POOL RADIUS (m)',/,10X,'CONTAMINANT MASS FRACTION (kg C',
     $         '/kg mix), and TEMPERATURE (K)')
2280   FORMAT(/,5X,'The suggested LOWEST CONCENTRATION OF INTEREST ',
     $'(gas_lfl/2.)',/,5x,' is ',1pg13.5,
     $' kg/m**3.  Enter the desired value: ',$)
c
       RETURN
       END
####
```

```
      subroutine limit(func, x0, xinc, xhigh, xlow)
c
c*** subroutine to establish the upper and lower limits for ZBRENT
c
      implicit real*8(a-h,o-z), integer*4(i-n)
      rrr = 1.D0
      aflag = 1.D0
      bflag = 1.D0
      fc = func(x0)
100   continue
      if(aflag .eq. 1.D0) then
         aaa = x0 + xinc*rrr
              if(aaa .ge. xhigh) then
                      aaa = xhigh
                      aflag = 0.D0
              endif
         fa = func(aaa)
              if(sign(1.D0,fa)*sign(1.D0,fc) .lt. 0.D0) then
                      xhigh = aaa
                      xlow  = aaa - xinc*1.01D0
                      return
              endif
      endif
      if(bflag .eq. 1.D0) then
         bbb = x0 - xinc*rrr
              if(bbb .le. xlow) then
                      bbb = xlow
                      bflag = 0.D0
              endif
         fb = func(bbb)
              if(sign(1.D0,fb)*sign(1.D0,fc) .lt. 0.D0) then
                      xhigh = bbb + xinc*1.01D0
                      xlow  = bbb
                      return
              endif
      endif
      if(sign(1.D0,fa)*fb .gt. 0.D0) then
              rrr = rrr + 1.D0
              if(rrr .gt. 400.D0) then
                      write(6,*) 'limit1? rrr>',rrr
                      rrr = 0.
                      rrr = 1./rrr
                      write(6,*) rrr
                      stop 'limit1? rrr>400'
              endif
              goto 100
      endif
      xhigh = aaa
      xlow  = bbb
      return
      end
####
1 -- sys$degadis:limit.for                    11-OCT-1989 19:17:34
```

```
C.............................................................
C
C    SUBROUTINE FOR SOURCE EVALUATION WHEN NO GAS BLANKET
C       IS PRESENT.
C
      SUBROUTINE NOBL(timeout, reflag)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C

      include 'sys$degadis:DEGADIS1.dec'
C
      COMMON
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/ERROR/ STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl,swcl,swal,senl,srhl
     $/phlag/ check1,check2,again,check3,check4,check5
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
C
      REAL*8 ML,K
c
      LOGICAL REV
      logical check1,check2,again,check3,check4,check5
      logical reflag
      DATA REV/.TRUE./
      REAL*8 L
c
      data h/0./,Ri/0./
      data delt_min/0.5/
C
      DELTAT = (TEND - TSC1)/FLOAT(NOBLPT)
      if(deltat .lt. delt_min) then
            noblpt = int((tend-tsc1)/delt_min) +1
            deltat = (tend-tsc1)/float(noblpt)
            endif
C
      TO = TSC1
      IF(DELTAT .LT. 2.) GO TO 100
C
      WRITE(lunlog,1100)
      WRITE(lunlog,*) DELTAT
 1100 FORMAT(5X,'TIME INCREMENT USED ON LAST PORTION OF SOURCE CALC')
C
```

```
  100 CONTINUE
C
C     ESTABLISH LOOP TO FINISH SOURCE
C
      DO 110 I = 1,NOBLPT
C
      TIME = T0 + FLOAT(I)*DELTAT
      IF(I .EQ. NOBLPT) TIME = TEND
      L = 2.0*AFGEN2(PTIME,R1T,TIME,'R1T-BL')
      erate = AFGEN2(PTIME,ET,TIME,'ET-BL')
      flux = EraTe/(pi*L*L/4.D0)
      PWCP = AFGEN2(PTIME,PWC,TIME,'ET-BL')
      PWAP = (1.D0 - PWCP)/(1.D0 + HUMID)
      HPRIM = AFGEN2(PTIME,PENTH,TIME,'ET-BL')
      CALL SETDEN(PWCP, PWAP, HPRIM)
      RHOP = AFGEN2(PTIME,PRHO,TIME,'RH-BL')
      CCP = PWCP * RHOP
c
      qstar = CCP * k*ustar*alpha1*dellay/(dellay-1.)/phihat(RHOP,L)
      if(abs(flux/qstar) .gt. ernobl .and. reflag) then
             check3 = .true.
             timeout = time
             return
             end if
C
      call szf(flux,L,PWCP,sz,cclay,wclay,rholay)
      cc = cclay*dellay
      if(cc.gt.ccp) cc=ccp
c
c... Ensure that the SZF calculated values close the material balance
c
      twidth = pi*L/4.D0
      qqqq = erate/u0/z0*alpha1 * z0**alpha1 / twidth
      cc = qqqq/sz**alpha1
      if(cc.gt.ccp) cc=ccp
      sz = (qqqq/cc)**(1.D0/alpha1)
c
      call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enthalpy,temp)
c
      IF(Erate .LT. EMAX) GO TO 220
      EMAX = Erate
      RM = AFGEN2(PTIME,R1T,TIME,'R1T-BL')
      SZM = SZ
  220 CONTINUE
      RLIST = AFGEN2(PTIME,R1T,TIME,'R1T-BL')
      RMAX = dMAX1(RMAX,RLIST)
c
      WRITE(9,2000) TIME,RLIST,h,flux,SZ,yc,ya,rho,Ri,wc,wa,enthalpy,temp
c
      if(i.eq.3  .and.  check4) goto 500      ! steady
C
```

```
  110   CONTINUE
        RETURN
c
c
  500   continue                   ! steady state completion
        outcc = cc
        swcl = wc
        swal = wa
        senl = enthalpy
        srhl = rho
        outsz = sz
        outl  = 2. * rlist
        outb  = pi * rlist**2 /outl/2.
        return
 2000   format(1pg16.9,1x,1pg16.9,<iout_src-2>(1x,1pg13.6))
        END
####
```

```
C..............................................................
C       ......................................................
C
C       SUBROUTINES OB AND OBOUT ARE USED IN THE OBSERVER INTEGRATIONS
C               OVER THE SOURCE.
C
        SUBROUTINE OB(time,Y,D,PRMT)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
        include 'sys$degadis:DEGADIS2.dec'
c
        COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
     $/phicom/ iphifl,dellay
c
        REAL*8 K,ML
        logical flag
C
        DIMENSION Y(1),D(1),PRMT(1)
        INTEGER HWIDTH,Mrate,Crate,BDArate,Hrate
        DATA HWIDTH/1/,Mrate/2/,Crate/3/,BDArate/4/,Hrate/5/
C
C*** PASS TO IN PRMT(6)
C
        flag = isofl.eq. 1 .or. ihtfl.eq. 0
c
        TOL = PRMT(6)
        xup = prmt(7)
        XI = XIT(TIME,TOl)
        RG = AFGEN(RADG,TIME,'RADG')
        RLEN = PRMT(13)
C
        BIPR = 0.
        D(HWIDTH)= 0.
        D(Crate) = 0.
        D(Mrate) = 0.
        D(BDArate)= 0.
        D(Hrate) = 0.

c
c*** We must deal with the case when the observer sees no gas, but the
c*** integration is not over yet.  Since the derivatives have been set
c*** to zero above, no work on the derivatives is needed, but the values
c*** in PRMT(14-18) must be preserved.  Since these will be switched
c*** in OBOUT, switch them now so the old values will be retained.
```

```
c
        IF(ABS(XI) .GE. RG) then          ! use the last values
              PRMT(8)  = prmt(14)         ! cclay
              PRMT(9)  = prmt(15)         ! wclay
              PRMT(10) = prmt(16)         ! walay
              PRMT(11) = prmt(17)         ! enthlay
              PRMT(12) = prmt(18)         ! rholay
              RETURN
              endif
      BIPR = sqrt(RG*RG - XI*XI)
c
      UI = UIT(TIME,TOl)
c
      Q    = AFGEN(QSTR,TIME,'QSTR')
      wc   = AFGEN(srcwc,time,'srcwc')
      wa   = AFGEN(srcwa,time,'srcwa')
      enth = AFGEN(srcenth,time,'srcenth')
c
      wclay   = Y(Crate)/Y(Mrate)
      walay   = Y(BDArate)/Y(Mrate)
      if(.not.flag) enthlay = Y(Hrate)/Y(Mrate)
c
      call tprop(1,wclay,walay,enthlay,yc,ya,wm,temp,rholay,cp)
      cclay = wclay*rholay
c
      prmt(8) = cclay
      prmt(9) = wclay
      prmt(10)= walay
      prmt(11)= enthlay
      prmt(12)= rholay
c
      cc  = cclay*dellay
      rho = dellay*(rholay-rhoa) + rhoa
c
      szob = 0.01
      arg = Q*(xi-xup)/cc/(u0*z0/alpha1)
      if(xi.gt. xup .and. arg.gt.0.)
     1 szob = arg**(1./alpha1) * z0
c
      HEFF = GAMMAF/ALPHA1* SZOB
      RISTR=RIF(RHO,HEFF)
      PHI = PHIF(RISTR,0.)
      welay = dellay * K*USTAR* ALPHA1/PHI
C
      D(HWIDTH)= UI * BIPR / RLEN
      D(Crate) = D(HWIDTH)*RLEN * Q
      D(Mrate) = (Q/wc + rhoa*welay) * D(HWIDTH)*RLEN
      D(BDArate)= (Q*wa/wc + rhoa*welay/(1.+humid)) * D(HWIDTH)*RLEN
      if(flag) return
      D(Hrate) = Q * enth/wc * D(HWIDTH)*RLEN
C
```

```
  1000   FORMAT(' ?OB? -- Value of XI ',1pG13.4,'; Value of RG ',
     $   1pG13.4)
         RETURN
         END
c
c
         SUBROUTINE OBOUT( X, Y, DERY, IHLF, NDIM, PRMT)
C
         Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

         DIMENSION X(1), Y(1), DERY(1), PRMT(1)
C
         PRMT(14) = prmt(8)       ! cclay
         PRMT(15) = prmt(9)       ! wclay
         PRMT(16) = prmt(10)      ! walay
         PRMT(17) = prmt(11)      ! enthlay
         PRMT(18) = prmt(12)      ! rholay
         RETURN
         END
####
```

```
C..............................................................
C
C       FUNCTION PSI
C
C*** AS PER COLENBRANDER --
C
C*** THIS FUNCTION HAS BEEN DERIVED FROM BUSINGER,J.A.
C***  WORKSHOP ON MICROMETEOROLOGY, CHAPTER 2, HAUGEN,D.A. (ED.)
C***  AMERICAN METEOROLOGICAL SOCIETY.
C
        FUNCTION PSIF(Z,ML)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C

        include 'sys$degadis:DEGADIS1.dec'
c
        REAL*8 ML
C
        IF( ML ) 10,20,30
C
 10     A = (1.-15.*Z/ML)**.25
        PSIF = 2.*dLOG((1.+A)/2.) + dLOG((1.+A*A)/2.) - 2.*daTAN(A) +
     $ PI/2.
        RETURN
C
 20     PSIF = 0.
        RETURN
C
 30     PSIF = -4.7*Z/ML
        RETURN
        END
####
```

```
C.................................................................
C
C       SUBROUTINES FOR PSEUDO-STEADY STATE INTEGRATION.
C
        SUBROUTINE PSS(DIST,Y,DERY,PRMT)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
        include 'sys$degadis:DEGADIS2.dec/list'
c
        parameter (zero=1.D-10, rcrit=2.D-3)
c
        COMMON
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
c
        REAL*8 K,ML

        LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
        DIMENSION Y(1),DERY(1),PRMT(1)
        DATA rhouh/1/,SY2/2/,BEFF/3/,dh/4/,Mhi/5/,Mlow/6/
        INTEGER rhouh,SY2,BEFF,dh, Mhi, Mlow
C
C*** PRMT I/O SETUP
C***   I        VALUE            IN/OUT
C***  ----      -----            -----
C***   6          E               IN
C***   7          Cc             OUT
C***   8          Bb             OUT
C***   9        CON DERY(BEFF)    IN
C***  10        CON DERY(SZ)      IN
C***  11        NREC(I,1)        OUT -- STARTS OUTPUT COUNTER
C***  12        DIST             OUT
c***  13
c***  14        yc               out
c***  15        rho              out
c***  16        temp             out - if recorded
c***  17        gamma            out - if recorded
c***  18
c***  19
c***  20
c***  21        sz
c***  22        sz
```

```
C
        Erate = PRMT(6)
        Bb = Y(BEFF) - SQrtPI/2.*sqrt(Y(SY2))
c
c using the last value for Sz
c
        sz0 = prmt(22)
        sz = sz0
C
C*** MATERIAL BALANCE
C
        iii = 0
 100    Cc = Erate*ALPHA1/2./U0*(Z0/SZ)**ALPHA/SZ/Y(BEFF)
        cclay = cc/dellay
        call addheat(cclay,y(dh),rholay,temlay,cp)
        prod = dmax1( Y(rhouh)/rholay/prmt(18), zero)
        sz = ( prod )**(1./alpha1) * z0
        dif = abs(sz - sz0)/(abs(sz)+abs(sz0)+zero)
        if(dif .gt. rcrit) then
                sz0 = sz
                iii=iii+1
                if(iii .gt. 20) call trap(32)
                goto 100
                endif
        prmt(20) = rholay
        prmt(21) = sz
        HEFF = GAMMAF/ALPHA1*SZ

        call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enth,temp)
        call adiabat(0,wc,wa,yclay,ya,cclay,rholam,wml,enth,temlay)
c
        rit = 0.
        if(isofl.eq.0 .and. ihtfl.ne.0) then
                call addheat(cc,dellay*y(dh),rho,temp,cp)
                rit = rift(temp,heff)
                endif
        RISTR = RIF(RHO,HEFF)
        PHI = PHIF(RISTR,rit)
C
C*** CALCULATE DERIVATIVES
C
        DERY(BEFF) = 0.
        delrho = rho-rhoa
        IF(delrho .GT. delrhomin) DERY(BEFF) = PRMT(9)*sqrt(delrho/rhoa)
     $    *(SZ/Z0)**(.5 - ALPHA)
C
        DERY(SY2) = 8.*BETA/PI*Y(BEFF)**2 *
     $    (DELTA*SQPI02/Y(BEFF)) ** (1./BETA)
C
c
        heigh = heff*dellay
```

```
      yw = 1.-yclay-ya
      yw = min( max( yw, 0.D0 ), 1.D0)
      call surface(temlay,heigh,rholay,wml,cp,yw,watrte,qrte)
      if(temp.ge. tsurf .or. temlay.ge. tamb) qrte = 0.
      rhouhb = rholay* prmt(18) * (sz/z0)**alpha1 * Y(beff)
      d_rhouhb = prmt(19)*y(beff)/phi
      dERY(dh) = (qrte*Y(beff)/dellay - Y(dh)*d_rhouhb)/rhouhb
      dERY(rhouh) = (d_rhouhb-Y(rhouh)*DERY(beff))/Y(beff)
c
c*** Calculate the derivative for the total mass above the UFL and LFL
c
      gamma = (rho-rhoa)/cc                         ! gamma
      if(check4) then
        DERY(mlow) = 0.
        DERY(mhi)  = 0.
        if( isofl.eq.1 .or. ihtfl.eq.0 ) then
              call adiabat(2,aa,dd,gas_ufl,ee,chi ,gg,hh,pp,oo)
              call adiabat(2,aa,dd,gas_lfl,ee,clow,gg,hh,pp,oo)
        else
              call adiabat(-2,aa,dd,gas_ufl,ee,chi ,gg,hh,gamma,oo)
              call adiabat(-2,aa,dd,gas_lfl,ee,clow,gg,hh,gamma,oo)
        endif
              gamhi  = 2.D0* Cc * Bb * Sz / alpha1
      gammax = 2.D0* Cc * Sz * GAMMAF/alpha1 * (Bb+sqrtpi/2.D0*sqrt(Y(Sy2)))

        if(cc.gt.clow) then
              wlow = Dlog(cc/clow)
              gamlow = gaminc(1.D0/alpha1, wlow ) * gamhi
          DERY(mlow)= gamlow + 2.D0*clow*sqrt(Y(Sy2))*Sz/alpha1*series(wlow)
          DERY(mlow)= DMIN1( DERY(mlow), gammax )
        endif

        if(cc.gt.chi) then
              whi  = Dlog(cc/chi )
              gamhi  = gaminc(1.D0/alpha1, whi  ) * gamhi
          DERY(mhi) = gamhi  + 2.D0*chi *sqrt(Y(Sy2))*Sz/alpha1*series(whi )
          DERY(mhi) = DMIN1( DERY(mhi ), gammax )
        endif
      endif
C
C*** RETURNED VALUES
C
      PRMT(7) = Cc
      PRMT(8) = Bb
      prmt(14)= yc
      prmt(15)= rho
      prmt(16)= temp
      prmt(17)= gamma
      RETURN
      END
####
```

```
C.............................................................
C
C       SUBROUTINE PSSOUT
C
        SUBROUTINE PSSOUT(X,Y,D,IHLF,NDIM,PRMT)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        include 'sys$degadis:DEGADIS2.dec'
c
        parameter (npss=9, zero=1.e-10)
c
        COMMON
     $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/STP/STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/STOPIT/TSTOP
c
        REAL*8 K,ML
        LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
        DIMENSION Y(1),D(1),PRMT(1),BKSP(npss),OUT(npss),CURNT(npss)
C
C*** OUTPUT PARAMETERS
C
C*** FROM PSS             OUTPUT TO MODEL
C*** --------             ---------------
C***    X                      DIST
C***    PRMT(7)                Cc
C***    Y(1)                   SZ
C***    Y(2)                   SY2
C***    PRMT(8)                B
C***    PRMT(13)               TO(I)
c***    prmt(14)               yc
c***    prmt(15)               rho
c***    prmt(16)               temp
c***    prmt(17)               gamma
c
        ERM = 0.
        TSL = TS(PRMT(13),X)
        prmt(22) = prmt(21)
        IF(PRMT(11) .NE. 0.) GO TO 90
C
C*** STARTUP FOR THE OUTPUT ROUTINE
C
        RII = -100./STPP
        RI = 0.
        CURNT(1) = X
        curnt(2) = prmt(14)     ! yc
        CURNT(3) = PRMT(7)      ! cc
        curnt(4) = prmt(15)     ! rho
        curnt(5) = prmt(17)     ! gamma
        curnt(6) = prmt(16)     ! temp
        CURNT(7) = prmt(21)     ! sz
        CURNT(8) = sqrt(Y(2))   ! sy2
        CURNT(9) = PRMT(8)      ! b
        IF(prmt(8) .LE. 0.) CALL trap(16)
   90   CONTINUE
C
C*** STOP INTEGRATION WHEN THE HALF WIDTH B < 0.
C
        IF( PRMT(8) .LE. 0.) GO TO 1000
C
C*** SET THE CURRENT AND PREVIOUS RECORD
C
        DO 100 II=1,npss
  100   BKSP(II) = CURNT(II)
C
        CURNT(1) = X
        curnt(2) = prmt(14)     ! yc
        CURNT(3) = PRMT(7)      ! cc
        curnt(4) = prmt(15)     ! rho
        curnt(5) = prmt(17)     ! gamma
        curnt(6) = prmt(16)     ! temp
        CURNT(7) = prmt(21)     ! sz
        CURNT(8) = sqrt(Y(2))   ! sy2
        CURNT(9) = PRMT(8)      ! b
C
C*** STOP INTEGRATION AND GET A NEW OBSERVER WHEN Cc<CcLOW
C
        IF(PRMT(7).GT.CcLOW .OR. TSL.LT.TSTOP) GO TO 95
                if(prmt(11) .lt. 5.) then       ! guarantee 5 records
                        erm = oulp              ! force output
                        goto 95
                        endif
        AGAIN = .TRUE.
        TSTOP = TSL
        GO TO 1000
   95   CONTINUE
C
c*** Check the error criteria's
c
        RI = RI + 1.
        II = 2                  ! skip DIST,YC; 1 and 2
  110   II = II + 1
        ER1 = ABS( (CURNT(II)-BKSP(II))/(CURNT(II)+zero) )
        ER2 = ABS( (CURNT(II)-OUT(II))/(CURNT(II)+zero) )
        ERM = dMAX1(ER1,ER2,ERM)
        IF(II .EQ. 3) II = II + 1       ! skip RHO;4
        IF(II .EQ. 7) II = II + 1       ! skip SY;8
        IF(II .EQ. 5) II = II + 1       ! skip TEMP;6
        IF(II .LT. npss) GO TO 110
C
C*** RECORD POINT IF ODLP IS EXCEEDED OR 80 METERS SINCE LAST RECORD
C*** RECORD FIRST POINT
C
        DX = CURNT(1) - OUT(1)
        IF( RI.NE.1. .AND. ERM.LT.ODLP .AND. DX.LE.ODLLP) RETURN
C
C*** IF THE NEXT INTEGRATION AFTER A POINT IS RECORDED VIOLATES THE
C*** ERROR BOUND, THE CURRENT POINT MUST BE RECORDED. OTHERWISE, THE
C*** LAST POINT TO SATISFY THE ERROR LIMITS IS RECORDED.
C
        DO 120 II=1,npss
        IF(RI .EQ. RII+1.) BKSP(II) = CURNT(II)
  120   OUT(II) = BKSP(II)
C
        RI = RII
        PRMT(11) = PRMT(11) + 1.
C
        WRITE(9,*) (OUT(II),II=1,npss)
        RETURN
C
 1000   CONTINUE
C
C*** STOP INTEGRATION
C
        PRMT(12) = X
C
        IF(CURNT(1) .EQ. OUT(1)) GO TO 130
C
        PRMT(11) = PRMT(11) + 1.
        WRITE(9,*) (CURNT(II),II=1,npss)
C
  130   CONTINUE
        PRMT(5) = 1.
        RETURN
        END
####
```

```
C..............................................................
C
C       SUBROUTINE PSSOUT
C
        SUBROUTINE PSSOUT(X,Y,DERY,IHLF,NDIM,PRMT)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        include 'sys$degadis:DEGADIS2.dec/list'
c
        parameter (npss=9, zero=1.e-10)
c
        COMMON
     $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/STP/STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_fl/ cflag,clfl,cufl
     $/ALP/ALPHA,alpha1
C
        logical cflag
        LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
        REAL*8 ML,K
C
        DIMENSION Y(1),DERY(1),PRMT(1)
        dimension BKSP(npss),OUT(npss),CURNT(npss)
C
C*** OUTPUT PARAMETERS
C
C*** FROM PSS              OUTPUT TO MODEL
C*** --------              --------------
C***    X                      DIST
C***    PRMT(7)                Cc
C***    Y(1)                   SZ
C***    Y(2)                   SY2
C***    PRMT(8)                B
C
        ERM = 0.
        prmt(22) = prmt(21)
C
        IF(PRMT(11) .NE. 0.) GO TO 90
C
C*** STARTUP FOR THE OUTPUT ROUTINE
C
        RII = -100./STPP
        RI = 0.
        CURNT(1) = X
        CURNT(2) = PRMT(14)     ! yc
        CURNT(3) = prmt(7)      ! cc
        CURNT(4) = prmt(15)     ! rho
```

```
      CURNT(5) = PRMT(17)     ! gamma
      curnt(6) = prmt(16)     ! temp
      curnt(7) = prmt(8)      ! b
      curnt(8) = prmt(21)     ! sz
      curnt(9) = sqrt(Y(2))   ! sy2
C
   90 CONTINUE
C
C*** STOP INTEGRATION WHEN THE HALF WIDTH B < 0.
C
      IF( PRMT(8) .LE. 0.) GO TO 1000
C
C*** STOP INTEGRATION when Cc<CcLOW
C
      IF(PRMT(7).GT.CcLOW) GO TO 95
              if(prmt(11) .lt. 5.) then        ! force output
                      erm = odlp
                      goto 95
                      endif
      AGAIN = .TRUE.
      GO TO 1000
   95 CONTINUE
C
C*** SET THE CURRENT AND PREVIOUS RECORD
C
      DO 100 II=1,npss
  100 BKSP(II) = CURNT(II)
C
      CURNT(1) = X
      CURNT(2) = PRMT(14)     ! yc
      CURNT(3) = prmt(7)      ! cc
      CURNT(4) = prmt(15)     ! rho
      CURNT(5) = PRMT(17)     ! gamma
      curnt(6) = prmt(16)     ! temp
      curnt(7) = prmt(8)      ! b
      curnt(8) = prmt(21)     ! sz
      curnt(9) = sqrt(Y(2))   ! sy2
C
      RI = RI + 1.
      II = 1
  110 II = II + 1
      ER1 = ABS( (CURNT(II)-BKSP(II))/(CURNT(II)+zero) )
      ER2 = ABS( (CURNT(II)-OUT(II))/(CURNT(II)+zero) )
      ERM = dMAX1(ER1,ER2,ERM)
      IF(II .EQ. 3) II = 6    ! skip density,gamma,temp
      IF(II .LT. npss-1) GO TO 110    ! skip sy
C
C*** RECORD POINT IF ODLP IS EXCEEDED OR 80 METERS SINCE LAST RECORD
C*** RECORD FIRST POINT
C
      DX = CURNT(1) - OUT(1)
      IF( RI.NE.1. .AND. ERM.LT.ODLP .AND. DX.LE.ODLLP) RETURN
C
C*** IF THE NEXT INTEGRATION AFTER A POINT IS RECORDED VIOLATES THE
C*** ERROR BOUND, THE CURRENT POINT MUST BE RECORDED. OTHERWISE, THE
C*** LAST POINT TO SATISFY THE ERROR LIMITS IS RECORDED.
C
      DO 120 II=1,npss
      IF(RI .EQ. RII+1.) BKSP(II) = CURNT(II)
  120 OUT(II) = BKSP(II)
C
      RI = RII
      PRMT(11) = PRMT(11) + 1.
C
      call ssout(out)
      RETURN
C
 1000 CONTINUE
C
C*** STOP INTEGRATION
C
      PRMT(12) = X
C
      IF(RI .EQ. 0.) CALL trap(16)
C
      IF(CURNT(1) .EQ. OUT(1)) GO TO 130
C
      PRMT(11) = PRMT(11) + 1.
      call ssout(out)
C
  130 CONTINUE
      PRMT(5) = 1.
C
      RETURN
      END
####
```

```
C.........................................................
C
C       RICHARDSON NUMBER (RI*)
C
        FUNCTION RIF(RHOG,HEFF)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        COMMON
     $ /PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
C
        REAL*8 ML,K
C
        RIF = G*(RHOG-RHOA)/RHOA*HEFF/USTAR/USTAR
C
        RETURN
        END
C
C.........................................................
C
C       RICHARDSON NUMBER (RIt)
C
        FUNCTION RIFt(temp,HEFF)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        COMMON
     $ /PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/alp/ alpha,alpha1
C
        REAL*8 ML,K
C
        wind = u0*(heff/z0)**alpha
        RIFt = dmax1(G*(tsurf-temp)/temp*HEFF/USTAR/wind,0.D0)
C
        RETURN
        END
C
C...............................................................
C
C       PHI FUNCTION
C
        FUNCTION PHIF(RI,rit)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

c
```

```
      common /phicom/ iphifl,dellay
c
      phif= 0.
      goto(10,1000,2000,3000,4000,9000),iphifl
      goto 9000
c
   10 IF(RI) 100,200,300
c
  100 PHIF = 0.74/(1. + 0.65*ABS(RI)**.6)
      RETURN
c
  200 PHIF = 0.74
      RETURN
c
  300 PHIF = 0.74 + 0.25*(RI)**0.7 + 1.2E-7*RI*RI*RI
      RETURN
c
c
 1000 IF(RI) 1100,1200,1300
c
 1100 PHIF = 0.88/(1. + 0.65*ABS(RI)**.6)
      RETURN
c
 1200 PHIF = 0.88
      RETURN
c
 1300 PHIF = 0.88 + 9.9e-2*(RI)**1.04 + 1.4E-25*RI**5.7
      RETURN
c
c
c
 2000 corr1 = 0.25* rit**.666666 + 1.
      corr = sqrt(corr1)
      riw = ri/corr1
      IF(RI) 2100,2200,2300
c
 2100 PHIF = 0.88/(1. + 0.65*ABS(RIw)**.6)/corr
      RETURN
c
 2200 PHIF = 0.88/corr
      RETURN
c
 2300 PHIF = (0.88 + 9.9e-2*(RIw)**1.04 + 1.4E-25*RIw**5.7)/corr
      RETURN
c
c
 3000 corr1 = 0.25* rit**.666666 + 1.
      corr = sqrt(corr1)
      riw = ri/corr1
      IF(RI) 3100,3200,3300
c
```

```
 3100   PHIF = 0.88/corr
        RETURN
c
 3200   PHIF = 0.88/corr
        RETURN
c
 3300   PHIF = (0.88 + 9.9e-2*(RIw)**1.04 + 1.4E-25*RIw**5.7)/corr
        RETURN
c
c
 4000   PHIF = 0.88
        RETURN
c
c
 9000   call trap(29)
        return
        END
c
c
c..............................................................
c
c
        function phihat(rho,fetch)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

c
        common
     $/parm/ u0,z0,zr,ml,ustar,k,g,rhoe,rhoa,delta,beta,gammaf,cclow
     $/alp/ alpha,alpha1
     $/phicom/ iphifl,dellay
c
        real*8 k,ml
c
        data phic/3.1/
c
        if(rho .le. rhoa) then
                phihat = 0.88
                return
                endif
c
        pow = 1./alpha1
        p1 = 1.04/alpha1
        p1i = 1.D0/p1
        p2 = 1. + p1i
        p3 = (alpha - 0.04)/1.04
        p4 = (1.08 - alpha)/1.04
        Ci = g*(rho-rhoa)/rhoa*z0/ustar**2*gammaf/alpha1
        Ci =Ci* (k*ustar*alpha1**2 /u0/z0/ phic*dellay/(dellay-1.)) ** pow
c
        Ri = Ci*fetch**pow
```

```
      zzz =  - 0.099*Ri**1.04/0.88
c
      if(abs(zzz) .lt. 1.) then
            phihat = 0.88/gseries(1.D0, p1i, p2, zzz)
      else
            zzz = 1.D0/zzz
            phihat = 0.88/(-zzz/(1.-p1)*gseries(1.D0, -p3, p4, zzz)
     .        + gamma(p1i)*gamma(p4)/(p1-1.)* (-zzz)**(p1i))
      endif
c
      return
      end


c
c------------------------------------------------------------
c
c     function to calculate the hypergeometric series
c
      function gseries( aaa, bbb, ccc, zzz)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      data crit/1.D-5/
c
c*** check "zzz" to see if it is greater than 0.999.  If so, calculate
c     "gseries" for "zzz"=0.999 to avoid excessive execution times.
c
      if( abs(zzz) .gt. 0.999) then
            zzzz = sign(0.99900, zzz)
      else
            zzzz = zzz
      endif

      term = aaa*bbb/ccc*zzzz
      sumo = 1.D0 + term

      do k = 1,100000
         term = term * (aaa+k)*(bbb+k)/(ccc+k) * zzzz/(k+1)
         sum = sumo + term
         if( abs(term/sum) .le. crit) goto 1000
         sumo = sum
      enddo

      write(6,*) 'GSERIES? did not converge in RIPHIF'
      stop

 1000 gseries = (sum + sumo)/2.D0
      return
      end
####
```

```
C       .........................................................
C
C       SUBROUTINE RKGST
C
C.................................................................
C
C       This routine was originally supplied by Digital Equipment
C       Corporation as part of the Scientific Subroutine Package
C       available for RT-11 as part of the Fortran Enhancement
C       Package.  It was upgraded for use as the integration
C       routine in this package.
C
C.................................................................
C
C
C       PURPOSE
C             TO SOLVE A SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL
C             EQUATIONS WITH GIVEN INITIAL VALUES.
C
C       USAGE
C             CALL RKGST (PRMT,Y,DERY,NDIM,IHLF,FCT,OUTP,AUX)
C             PARAMETERS FCT AND OUTP REQUIRE AN EXTERNAL STATEMENT.
C
C       DESCRIPTION OF PARAMETERS
C
C       PRMT    AN INPUT AND OUTPUT VECTOR WITH DIMENSION GREATER
C               OR EQUAL TO 5, WHICH SPECIFIES THE PARAMETERS OF
C               THE INTERVAL AND OF ACCURACY AND WHICH SERVES FOR
C               COMMUNICATION BETWEEN SUBROUTINES OUTP AND FCT
C               (FURNISHED BY THE USER) AND SUBROUTINE RKGST.
C               EXCEPT PRMT(5) THE COMPONENTS ARE NOT DESTROYED
C               BY SUBROUTINE RKGST AND THEY ARE:
C       PRMT(1) LOWER BOUND OF THE INTERVAL (INPUT),
C       PRMT(2) UPPER BOUND OF THE INTERVAL (INPUT),
C       PRMT(3) INITIAL INCREMENT OF THE INDEPENDENT VARIABLE
C               (INPUT),
C       PRMT(4) UPPER ERROR BOUND (INPUT). IF RELATIVE ERROR IS
C               GREATER THAN PRMT(4), INCREMENT GETS HALVED.
C               IF RELATIVE ERROR LESS THAN PRMT(4)*EXPAND,
C               INCREMENT GETS DOUBLED.
C               THE USER MAY CHANGE PRMT(4) BY MEANS OF HIS
C               OUTPUT SUBROUTINE.
C       PRMT(5) MAXIMUM STEP SIZE ORDER OF MAGNITUDE (INPUT).
C               SUBROUTINE RKGST INITIALIZES
C               PRMT(5)=0. IF THE USER WANTS TO TERMINATE
C               SUBROUTINE RKGST AT ANY OUTPUT POINT, HE HAS TO
C               CHANGE PRMT(5) TO NON-ZERO BY MEANS OF SUBROUTINE
C               OUTP. FURTHER COMPONENTS OF VECTOR PRMT ARE
C               FEASIBLE IF ITS DIMENSION IS DEFINED GREATER
C               THAN 5. HOWEVER SUBROUTINE RKGST DOES NOT REQUIRE
C               AND CHANGE THEM. NEVERTHELESS THEY MAY BE USEFUL
C               FOR HANDING RESULT VALUES TO THE MAIN PROGRAM
```

```
C             (CALLING RKGST) WHICH ARE OBTAINED BY SPECIAL
C             MANIPULATIONS WITH OUTPUT DATA IN SUBROUTINE OUTP.
C        Y    INPUT VECTOR OF INITIAL VALUES.  (DESTROYED)
C             LATER, Y IS THE RESULTING VECTOR OF DEPENDENT
C             VARIABLES COMPUTED AT INTERMEDIATE POINTS X.
C        DERY INPUT VECTOR OF ERROR WEIGHTS.  (DESTROYED)
C             ERROR WEIGHTS ARE CENTERED AT ONE. IF ONE PARA-
C             METER NEEDS A TIGHTER ERROR CRITERIA,THE WEIGHT IS
C             GREATER THAN ONE. IF A PARAMETER NEED NOT BE DETER-
C             MINED SO PRECISELY,THE WEIGHT SHOULD BE LESS
C             THAN ONE.IN OTHER WORDS,
C                   ERROR CRITERIA(I) = PRMT(4) / WEIGHT(I)
C             WHERE I IS THE SUBSCRIPT OF A DEPENDENT VARIABLE.
C             LATER, DERY IS THE VECTOR OF DERIVATIVES, WHICH
C             BELONG TO FUNCTION VALUES Y AT A POINT X.
C        NDIM AN INPUT VALUE, WHICH SPECIFIES THE NUMBER OF
C             EQUATIONS IN THE SYSTEM.
C        IHLF AN OUTPUT VALUE, WHICH SPECIFIES THE NUMBER OF
C             BISECTIONS OF THE INITIAL INCREMENT. IF IHLF BE-
C             COMES GREATER THAN 10, SUBROUTINE RKGST RETURNS THE
C             ERROR MESSAGE IHLF=11 INTO MAIN PROGRAM. ERROR
C             MESSAGE IHLF=12 OR IHLF=13 APPEARS IN CASE
C             PRMT(3)=0 OR IN CASE SIGN(PRMT(3)).NE.SIGN(PRMT(2)-
C             PRMT(1)) RESPECTIVELY.
C        FCT  THE NAME OF AN EXTERNAL SUBROUTINE USED. THIS
C             SUBROUTINE COMPUTES THE RIGHT HAND SIDES DERY OF
C             THE SYSTEM TO GIVEN VALUES X AND Y. ITS PARAMETER
C             LIST MUST BE X,Y,DERY,PRMT. SUBROUTINE FCT SHOULD
C             NOT DESTROY X AND Y.
C        OUTP THE NAME OF AN EXTERNAL OUTPUT SUBROUTINE USED.
C             ITS PARAMETER LIST MUST BE X,Y,DERY,IHLF,NDIM,PRMT.
C             NONE OF THESE PARAMETERS (EXCEPT, IF NECESSARY,
C             PRMT(4),PRMT(5),...) SHOULD BE CHANGED BY
C             SUBROUTINE OUTP. IF PRMT(5) IS CHANGED TO NON-ZERO,
C             SUBROUTINE RKGST IS TERMINATED.
C        AUX  AN AUXILIARY STORAGE ARRAY WITH 8 ROWS AND NDIM
C             COLUMNS.
C
C     REMARKS
C
C     THE PROCEDURE TERMINATES AND RETURNS TO CALLING PROGRAM, IF
C     (1) MORE THAN 10 BISECTIONS OF THE INITIAL INCREMENT ARE
C             NECESSARY TO GET SATISFACTORY ACCURACY (ERROR MESSAGE
C             IHLF=11),
C     (2) INITIAL INCREMENT IS EQUAL TO 0 OR HAS WRONG SIGN
C             (ERROR MESSAGES IHLF=12 OR IHLF=13),
C     (3) THE WHOLE INTEGRATION INTERVAL IS WORKED THROUGH,
C     (4) SUBROUTINE OUTP HAS CHANGED PRMT(5) TO NON-ZERO.
C
C     SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C     THE EXTERNAL SUBROUTINES FCT(X,Y,DERY,PRMT) AND
```

```
C       OUTP(X,Y,DERY,IHLF,NDIM,PRNT) MUST BE FURNISHED BY THE USER.
C
C       METHOD
C       EVALUATION IS DONE BY MEANS OF FOURTH ORDER RUNGE-KUTTA
C       FORMULAE IN THE MODIFICATION DUE TO GILL. ACCURACY IS
C       TESTED COMPARING THE RESULTS OF THE PROCEDURE WITH SINGLE
C       AND DOUBLE INCREMENT.
C       SUBROUTINE RKGST AUTOMATICALLY ADJUSTS THE INCREMENT DURING
C       THE WHOLE COMPUTATION BY HALVING OR DOUBLING. IF MORE THAN
C       10 BISECTIONS OF THE INCREMENT ARE NECESSARY TO GET
C       SATISFACTORY ACCURACY, THE SUBROUTINE RETURNS WITH
C       ERROR MESSAGE IHLF=11 INTO MAIN PROGRAM.
C       TO GET FULL FLEXIBILITY IN OUTPUT, AN OUTPUT SUBROUTINE
C       MUST BE FURNISHED BY THE USER.
C       FOR REFERENCE, SEE
C       RALSTON/WILF, MATHEMATICAL METHODS FOR DIGITAL COMPUTERS,
C       WILEY, NEW YORK/LONDON, 1960, PP.110-120.
C
C       SOME NOTES ON THE PROGRAM/RALSTON AND WILF
C
C       AUX
C       ---
C
C       AUX(1,I) -- CURRENT VALUE OF Y
C       AUX(2,I) -- CURRENT VALUE OF Y'
C       AUX(3,I) -- LAST GOOD VALUES OF Q
C       AUX(4,I) -- Y AFTER ONE RK STEP H
C       AUX(5,I) -- Y AFTER ONE OR TWO RK STEPS OF H/2.
C       AUX(6,I) -- CURRENT VALUES OF Q
C       AUX(7,I) -- Y' AFTER ONE OR TWO RK STEPS OF H/2.
C       AUX(8,I) -- 2/15 * WEIGHTS
C
C       A(4),B(4),C(4)
C       --------------
C
C       Y  = Y    + A *(K   - B *Q    )
C        i    i-1    i   i     i  i-1
C
C       Q  = Q    + 3*(A *(K    - B *Q    ) - C *K
C        i    i-1      i   i      i  i-1      i  i
C
C       FOR VALUES OF I BETWEEN 1 AND 4, AND FOR VALUES OF K AS FOLLOWS
C
C       K  = H * F(X ,Y )
C        1          0  0
C
C       K  = H * F(X +H/2,Y )
C        2          0      1
C
C       K  = H * F(X +H/2,Y )
C        3          0      2
```

```
C
C       K  = H * F(X +H,Y )
C        4         0   3
C
C       RELATIVE ERROR
C       -------- -----
C
C       AS PER RICHARDSON QUOTED IN RALSTON/WILF (P117),
C
C       ABS ERROR = WEIGHT/15*ABS(Y2 - Y1)
C
C       THEN, RELATIVE ERROR
C
C       REL ERROR = WEIGHT*2/15*ABS(Y2 - Y1)/SUM
c
c       where SUM = ABS(Y2 + Y1)
C
C       The solution tries to use SUM=abs(y2+y1) first. If this is zero,
C       then SUM=.25*ABS(Y1) is used since y1 and y2 must be oposite in
C       sign with equal magnitude. If this
C       quantity is zero as well, the values y2 and y1 both must be zero;
C       therefore, the difference is also zero which satisfies the
C       error criteria.
C
C       ..............................................................
C
        SUBROUTINE RKGST(PRMT,Y,DERY,NDIM,IHLF,FCT,OUTP,AUX)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        DIMENSION Y(1),DERY(1),AUX(8,NDIM),A(4),B(4),C(4),PRMT(1)
C
        DATA ERRSET/1./
C
C
        DO 10 I=1,NDIM
   10   AUX(8,I)=.133333333333333333D0 * DERY(I)
        X=PRMT(1)
        XEND=PRMT(2)
        H=PRMT(3)
        stpmin = abs(h/1024.D0)
        stpmax = abs(prmt(5))
        PRMT(5)=0.D0
        CALL FCT(X,Y,DERY,PRMT)
C
C*** ERROR TEST
C
        IF(H*(XEND-X))380,370,20
C
C*** PREPARATIONS FOR RUNGE-KUTTA METHOD
```

```
C
   20 A(1)=.5D0
      A(2)= 1.D0 - DSQRT( 0.5D0 )
      A(3)= 1.D0 + DSQRT( 0.5D0 )
      A(4)= 1.D0/6.D0
      B(1)=2.D0
      B(2)=1.D0
      B(3)=1.D0
      B(4)=2.D0
      C(1)=.5D0
      C(2)= A(2)
      C(3)= A(3)
      C(4)=.5D0
C
C*** PREPARATIONS OF FIRST RUNGE-KUTTA STEP
C
      DO 30 I=1,NDIM
      AUX(1,I)=Y(I)
      AUX(2,I)=DERY(I)
      AUX(3,I)=0.D0
   30 AUX(6,I)=0.D0
      IREC=0
      H=H+H
      IHLF=-1
      ISTEP=0
      IEND=0
C
C*** START OF A RUNGE-KUTTA STEP
C*** STEP = 2 * SPECIFIED STEP
C
   40 IF((X+H-XEND)*H)70,60,50
   50 H=XEND-X
   60 IEND=1
C
C*** RECORDING OF INITIAL VALUES OF THIS STEP
C
   70 CALL FCT(X,Y,DERY,PRMT)
      CALL OUTP(X,Y,DERY,IREC,NDIM,PRMT)
      IF(PRMT(5))400,80,400
   80 ITEST=0
   90 ISTEP=ISTEP+1
C
C*** START OF INNERMOST RUNGE-KUTTA LOOP
C
      J=1
  100 AJ=A(J)
      BJ=B(J)
      CJ=C(J)
      DO 110 I=1,NDIM
      R1=H*DERY(I)
      R2=AJ*(R1-BJ*AUX(6,I))
      Y(I)=Y(I)+R2
      R2=R2+R2+R2
  110 AUX(6,I)=AUX(6,I)+R2-CJ*R1
      IF(J-4)120,150,150
  120 J=J+1
      IF(J-3)130,140,130
  130 X=X+H/2.
  140 CALL FCT(X,Y,DERY,PRMT)
      GOTO 100
C
C*** END OF INNERMOST RUNGE-KUTTA LOOP
C
C*** TEST OF ACCURACY
C
  150 IF(ITEST)160,160,200
C
C*** IN CASE ITEST=0 THERE IS NO POSSIBILITY FOR TESTING OF ACCURACY
C*** IF(ITEST=0) RK STEP JUST PERFORMED WAS FOR TWICE THE SPECIFIED STEP
C
  160 DO 170 I=1,NDIM
  170 AUX(4,I)=Y(I)
      ITEST=1
      ISTEP=ISTEP+ISTEP-2
  180 IHLF=IHLF+1
      X=X-H
      H=H/2.
      DO 190 I=1,NDIM
      Y(I)=AUX(1,I)
      DERY(I)=AUX(2,I)
  190 AUX(6,I)=AUX(3,I)
      GOTO 90
C
C*** IN CASE ITEST=1 TESTING OF ACCURACY IS POSSIBLE ONLY IF EACH
C*** HALF OF THE INTERVAL IS DONE(I.E.,IFF ISTEP IS EVEN)
C
  200 IMOD=ISTEP/2
      IF(ISTEP-IMOD-IMOD)210,230,210
  210 CALL FCT(X,Y,DERY,PRMT)
      DO 220 I=1,NDIM
      AUX(5,I)=Y(I)
  220 AUX(7,I)=DERY(I)
      GOTO 90
C
C*** ORIGINAL VERSION; absolute error
C
C     COMPUTATION OF TEST VALUE DELT
C 230 DELT=0.                  !Good so far
C     DO 240 I=1,NDIM
C 240 DELT=DELT+AUX(8,I)*ABS(AUX(4,I)-Y(I))
C     IF(DELT-PRMT(4))280,280,250
C
```

```
C*** RELATIVE ERROR
C
  230    DELT = 0.
         DO 240 I=1,NDIM
         ARG = ABS(AUX(4,I) + Y(I))
         IF(ARG .EQ. 0.) arg = .25*ABS(AUX(4,I))
c--- if the next statement is true, aux(4,i)=y(i)=0.0; rer=0.
         IF(ARG .EQ. 0.) ARG = ERRSET
         RER = AUX(8,I)*ABS(AUX(4,I) - Y(I))/ARG
  240    DELT = dMAX1(DELT,RER)
         IF(DELT-PRMT(4)) 280,280,250
C
C*** ERROR IS TOO GREAT
C
  250    if(abs(h) .lt. stpmin) goto 360
         DO 270 I=1,NDIM
  270    AUX(4,I)=AUX(5,I)
C        WRITE(5,1200) DELT
C 1200   FORMAT(' ?RKGST? -- ERROR TOO GREAT',G13.5)
         ISTEP=ISTEP+ISTEP-4
         X=X-H
         IEND=0
         GOTO 180
C
C*** RESULT VALUES ARE GOOD
C
  280    CALL FCT(X,Y,DERY,PRMT)
         DO 290 I=1,NDIM
         AUX(1,I)=Y(I)
         AUX(2,I)=DERY(I)
         AUX(3,I)=AUX(6,I)
         Y(I)=AUX(5,I)
  290    DERY(I)=AUX(7,I)
         CALL FCT(X-H,Y,DERY,PRMT)
         CALL OUTP(X-H,Y,DERY,IHLF,NDIM,PRMT)
         IF(PRMT(5))400,300,400
  300    DO 310 I=1,NDIM
         Y(I)=AUX(1,I)
  310    DERY(I)=AUX(2,I)
         IREC=IHLF
         IF(IEND) 320,320,390
C
C*** INCREMENT GETS DOUBLED TO KEEP UP WITH HALF STEPPING
C
  320    IHLF=IHLF-1
         ISTEP=ISTEP/2
         H=H+H
C
C*** ALLOW THE PROGRAM TO EXPAND BEYOND ORIGINAL STEP SIZE SPECIFICATION
C*** UP TO THE MAXIMUM
C
```

```
        IF(abs(h).ge.stpmax) goto 40
C
C*** EXPAND H DUE TO LOW ERROR VALUE after Press et al.  Since a 'new'
c       step size is being chosen, reset IHLF and ISTEP to starting values.
c       STPMIN will still stop simulations which bisect the original interval
c       more than 10 times.
C
        if(delt.ne. 0.D0) then
           trial =  .9D0 * abs(h) * (abs(prmt(4)/delt)) ** .25D0
        else
           trial = 10.D0 * abs(h)
        endif
c       if(trial .ge. abs(h)) then
           h = sign( min(trial, stpmax), h)
           IHLF=-1
           ISTEP=0
c       endif
        GOTO 40
C
C*** RETURNS TO CALLING PROGRAM
C
  360   IHLF=11
        CALL FCT(X,Y,DERY,PRMT)
        GOTO 390
  370   IHLF=12
        GOTO 390
  380   IHLF=13
  390   CALL FCT(X,Y,DERY,PRMT)
        CALL OUTP(X,Y,DERY,IHLF,NDIM,PRMT)
  400   RETURN
        END
####
```

```
      PROGRAM SDEGADIS2
c
C***********************************************************************
C***********************************************************************
C***********************************************************************
C
C       Program description:
C
C       SDEGADIS2 is a simplification of DEGADIS2 which performs the downwind
C       dispersion portion of the calculation for a steady state source
C       described by DEGADIS1.
C
C
C       Program usage:
C
C       Consult Volume III of the Final Report to U. S. Coast Guard
C       contract DT-CG-23-80-C-20029 entitled "Development of an
C       Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures".
C
C       J. A. Havens
C       T. O. Spicer
C
C       University of Arkansas
C       Department of Chemical Engineering
C       Fayetteville, AR 72701
C
C       April 1985
C
C
C       This project was sponsored by the U. S. Coast Guard and the Gas
C       Research Institute under contract DT-CG-23-80-C-20029.
C
C
C       Disclaimer:
C
C       This computer code material was prepared by the University of
C       Arkansas as an account of work sponsored by the U. S. Coast Guard
C       and the Gas Research Institute.  Neither the University of Arkansas,
C       nor any person acting on its behalf:
C
C       a.  Makes any warranty or representation, express or implied,
C           with respect to the accuracy, completeness, or usefulness
C           of the information contained in this computer code material,
C           or that the use of any apparatus, method, numerical model,
C           or process disclosed in this computer code material may not
C           infringe privately owned rights; or
C
C       b.  Assumes any liability with respect to the use of, or for
C           damages resulting from the use of, any information,
C           apparatus, method, or process disclosed in this computer
C           code material.
```

```
C
C
C***********************************************************************
C***********************************************************************
C***********************************************************************
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS2.dec'
      include '($ssdef)'
C
      EXTERNAL PSS,PSSOUT,SSG,SSGOUT
C
      COMMON
     $/TITL/ TITLE
     $/GEN2/ DEN(5,igen)
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/ITI/ T1,TINP,TSRC,TOBS
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/ERROR/ SYOER,ERRP,SMXP,WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,
     $ WTRUH,WTDHG
     $/STP/ STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/com_ss/ ESS,SLEN,SWID,OUTCc,OUTSZ,OUTB,OUTL,swcl,swal,senl,srhl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_fl/ cflag,clfl,cufl
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     $/COM_SURF/ HTCUT
     ./oomsin/ oodist,avtime
C
C
C     DIMENSIONS/DECLARATIONS
C
      logical cflag

      real*4 tt1
      REAL*8 K,ML,L
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
c
      character*24 tinp,tsrc,tobs
      character*80 title(4)
      character*4 pound
      character*3 gas_name
C
      character*4 TR2,ER2,Sr3,SSD,TR3
```

```
C
      character*40 opnrup1
      character opnrup(40)
      EQUIVALENCE (OPNRUP(1),opnrup1)
c
      dimension prmt(22),y(6),dery(6),aux(8,6)
C
C.......................................................
C
C       DATA
C
      DATA POUND/'//  '/,POUNDN/-1.D-20/
C
      DATA TIME0/0./,NDIM/0/
C
      DATA TR2/'.TR2'/,ER2/'.ER2'/
      DATA Sr3/'.Sr3'/
      DATA SSD/'.SSD'/,TR3/'.TR3'/
C
C
C.....................................................
C
C       MAIN
C
      T1 = SECNDS(0.)
      istat = lib$date_time(TOBS)
      if(istat .ne. ss$_normal) stop'lib$date_time failure'
C
C*** GET THE FILE NAME FOR FILE CONTROL
C
      read(5,1135) nchar,opnrup
 1135 format(q,40a1)
C
      opnrup1 = opnrup1(1:nchar) // ER2(1:4)
      CALL ESTRT2ss(OPNRUP1)
C
C*** GET THE COMMON VARIABLES CARRIED FROM DEGADIS1
C
      opnrup1 = opnrup1(1:nchar) // tr2(1:4)
      CALL STRT2(OPNRUP1,H_masrte,CCP)
C
      opnrup1 = opnrup1(1:nchar) // sr3(1:4)
      OPEN(UNIT=8,TYPE='NEW',NAME=OPNRUP1,
     $ CARRIAGECONTROL='FORTRAN')
c
      cflag = isofl.eq. 1.or. ihtfl.eq. 0
C
      WRITE(8,1119)
      if(cflag) then
              WRITE(8,1116) gas_zsp,(100.*gas_lfl),(100.*gas_ufl)
              WRITE(8,1118)
```

```
        else
                WRITE(8,1115) gas_zsp,(100.*gas_lfl),(100.*gas_ufl)
                WRITE(8,1117)
                endif
        WRITE(8,1119)
c
C
C
 1115   FORMAT(1H0,1X,'Distance',2x,3x,'Mole',3x,
        1       'Concentration',1x,'Density',2x,3x,'Gamma',4x,
        1       'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
        1       'Width at z=',0pf6.2,' m to:',/,1x,11x,1x'Fraction',2x,
        1       11x,11x,11x,11x,3x,'Width',3x,11x,9x,
        1       2(1pg9.3,'mole%',1x))
 1116   FORMAT(1H0,1X,'Distance',2x,3x,'Mole',3x,
        1       'Concentration',1x,'Density',2x,
        1       'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
        1       'Width at z=',0pf6.2,' m to:',/,1x,11x,1x'Fraction',2x,
        1       11x,11x,11x,3x,'Width',3x,11x,9x,
        1       2(1pg9.3,'mole%',1x))
 1117   FORMAT(1H ,4X,'(m)',4x,11x,
        1       2(1X,'(kg/m**3)',1x),12x,4x,'(K)',
        1       5(8X,'(m)'))
 1118   FORMAT(1H ,4X,'(m)',4x,11x,
        1       2(1X,'(kg/m**3)',1x),4x,'(K)',
        1       5(8X,'(m)'))
 1119   FORMAT(1H )
C
C
C
C.............................................................
C
C       STEADY STATE CALCULATIONS
C
C
        opnrup1 = opnrup1(1:nchar) // ssd(1:4)
c       OPEN(UNIT=9,TYPE='NEW',NAME=OPNRUP1)
C
C
        AGAIN = .FALSE.
C
C
        L = OUTL
        B = OUTB
        SZO = OUTSZ
        Erate = ESS
        QSTRO = Erate/2.D0/L/B
        Cc = OUTCc
        wc = swcl
        wa = swal
        enth = senl
        rho = srhl
c
        if(cc.gt. ccp) then
              write(lunlog,1126) cc,ccp
 1126         format(/,' ',10('*****'),/,' cc: ',1pg13.5,' is greater',
        1      ' than ccp: ',1pg13.5,/,' ',10('*****'),/)
              cc =ccp
              endif
c
        ratio1= u0*z0/alpha1/ z0**alpha1 * cc /b/qstr0/l
        ratio = ratio1* sz0**alpha1 * (B + sqrtpi/2.D0*sy0er)
        if(ratio.lt. 1.D0) then
              sy0er = (1.D0/(RATIO1*sz0**alpha1) - b)*2.D0/sqrtpi
        else
              sz0 = (1.D0/((B+ sqrtpi/2.D0*sy0er)*ratio1))**(1.D0/alpha1)
        endif

        humsrc = (1.D0-wc-wa*(1.D0+humid))/wc
        call setden(wc,wa,enth)

        if(cflag) then
              call adiabat(2,twc,twa,gas_lfl,ya,clfl,r,w,t,tt)
              call adiabat(2,twc,twa,gas_ufl,ya,cufl,r,w,t,tt)
              endif

c
        cclay = cc/dellay
        call adiabat(0,wclay,walay,yclay,yalay,cclay,rholay,w,enthlay,t)
c
C
C*** let everyone know
C
        WRITE(lunlog,1170) L,B
        WRITE(lunlog,1180) QSTR0,SZ0
        write(lunlog,1185) wclay,walay,rholay,cclay,t
        write(lunlog,1186) wc,wa,rho,cc
c
 1170   FORMAT(' LENGTH: ',1pG13.5,' BEFF: ',1pG13.5)
 1180   FORMAT(' TAKEUP FLUX: ',1pG13.5,' SZ0: ',1pG13.5)
 1185   format(' wclay: ',1pg12.5,' walay: ',1pg12.5,
        1      ' rholay: ',1pg12.5,' Cclay: ',1pg12.5,/,
        1      ' temlay: ',1pg13.5)
 1186   format(' wc: ',1pg12.5,'  wa: ',1pg13.5,/,
        1      ' rho: ',1pg12.5,' Cc: ',1pg12.5)
C
C*** PREPARE FOR STEADY STATE INTEGRATION.
C
        PRMT(1) = L/2.D0
        PRMT(2) = 6.023D13
        PRMT(3) = STPP
```

```
      PRMT(4) = ERRP
      PRMT(5) = SMXP
      PRMT(6) = Erate
      PRMT(7) = Cc     ! OUTPUT
      PRMT(8) = B      ! OUTPUT
C
C*** PRMT(9) & PRMT(10) ARE CONSTANTS FOR D(SY) & D(SZ)
C
      PRMT(9) = Ce*sqrt(G*Z0/ALPHA1*GAMMAF) *GAMMAF/U0
      PRMT(10)= Z0**ALPHA*K*USTAR*ALPHA1 * ALPHA1/U0
C     PRMT(11)= NREC
C     PRMT(12)=
C     PRMT(13)=
      prmt(18)= u0*z0/alpha1
      prmt(19)= rhoa*k*ustar*alpha1
      prmt(20)= rholay
      prmt(21)= sz0
      prmt(22)= sz0
C
      Y(1) = rholay*prmt(18)*(SZ0/z0)**alpha1 ! rholay*ueff*heff
      Y(2) = SY0ER*sy0er
      Y(3) = B + sqrtpi/2.D0*sy0er
      Y(4) = 0.                         ! added heat
      Y(5) = 0.                         ! mass above UFL
      Y(6) = 0.                         ! mass above LFL
C
      DERY(1) = WTSZP
      DERY(2) = WTSYP
      DERY(3) = WTBEP
      dery(4) = wtdh
      dery(5) = 1.D0
      dery(6) = 1.D0
C
c     NDIM = 4
      ndim=6          ! to integrate the mass above LFL and UFL
C
      WRITE(lunlog,1130)
 1130 FORMAT('  Entering Integration Step -- B > 0. ')
C
C*** PERFORM INTEGRATION
C
      CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,PSS,PSSOUT,AUX)
C
      IF(IHLF .GE. 10) CALL trap(9,IHLF)
C
      NREC = INT(PRMT(11))
      WRITE(lunlog,1100)NREC
 1100 FORMAT(3X,'NUMBER OF RECORDS IN PSS = 'I10)
C
      IF(AGAIN) THEN
            Y(3) = Y(5)             ! mass above UFL
```

```
                Y(4) = Y(6)             ! mass above LFL
                GO TO 120
                ENDIF
C
C*** GAUSIAN COMPLETION OF THE INTEGRATION
C
C*** PSSOUT FORCES THE ABOVE INTEGRATION TO FINISH WHEN B<0 FOR THE
C***  FIRST TIME. THE STEP BEFORE THIS OCCURS IS RECORDED ON UNIT 7.
C***  THE STEP WHEN B GOES NEGATIVE IS CURRENTLY IN Y.
C
C*** THE CALCULATION METHOD CHANGES THE CURRENT VALUE OF SY TO A VALUE
C***  CALCULATED AS IF BEFF=SY RETAINING THE LAST VALUE OF Cc IN THE
C***  MATERIAL BALANCE.
C
        heat = Y(4)
        rholay = prmt(20)
        Cc = PRMT(7)
        rhouh = Y(1)
        SZ = ( rhouh/rholay/prmt(18) )**(1.D0/alpha1) * z0
        SYT = Erate*ALPHA1 *(Z0/SZ)**ALPHA/U0/SZ/Cc/SQRTPI
C
        XT = PRMT(12)
        XV = (SYT/RT2/DELTA)**(1.D0/BETA) - XT
C
C*** SET UP INTEGRATION FOR THE GAUSSIAN DISPERSION PHASE.
C
        do i=1,22
        prmt(i) = 0.D0
        enddo
c
        PRMT(1) = XT
        PRMT(2) = 6.023D23
        PRMT(3) = STPG
        PRMT(4) = ERRG
        PRMT(5) = SMXG
        PRMT(6) = Erate
        PRMT(7) = Cc            !-- OUTPUT
        PRMT(8) = XV
C       PRMT(9) = "BLANK"
C       PRMT(10)= "BLANK"
C       PRMT(11)= "BLANK"
C       PRMT(12)= DIST AT COMPLETION -- OUTPUT
C
        prmt(18) = u0*z0/alpha1
        prmt(19) = rhoa*k*ustar*alpha1
        prmt(20) = rholay
        prmt(21) = sz
        prmt(22) = sz
c
        Y(1) = rhouh
        Y(2) = heat
        Y(3) = Y(5)             ! mass above UFL
        Y(4) = Y(6)             ! mass above LFL
C
        DERY(1) = wtruh
        dery(2) = wtdhg
        dery(3) = 1.D0
        dery(4) = 1.D0
C
c NDIM = 2
        ndim=4          ! to integrate the mass above LFL and UFL
C
        WRITE(lunlog,1140)
 1140   FORMAT(' Entering Gaussian Stage of Integration ')
C
C*** PERFORM INTEGRATION
C
        CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,SSG,SSGOUT,AUX)
C
        IF(IHLF .GE. 10) CALL trap(10,IHLF)
C
        NREC = INT(PRMT(11))
C
  120   CONTINUE
c
c*** summarize the information about the mass above the LFL,UFL
c
        write(8,8000) 100.*gas_ufl,100.*gas_lfl,Y(4)-Y(3),Y(4)
 8000   format(//,' For the UFL of ',1pg13.5,' mole percent, and',
     $' the LFL of ',1pg13.5,' mole percent:',
     $//,' The mass of contaminant between the UFL and LFL is:'
     $,1pg13.5,' kg.',/,' The mass of contaminant above the LFL is: ',
     $1pg13.5,' kg.')
C
C
C..............................................................
C
c       CLOSE(UNIT=9)
        CLOSE(UNIT=8)
C
        opnrup1 = opnrup1(1:nchar) // tr3(1:4)
        CALL TRANS(OPNRUP1)
C
        tt1 = t1
        T1 = SECNDS(tT1)/60.
        WRITE(lunlog,4000) TOBS
        WRITE(lunlog,4010) T1
 4000   FORMAT(//,'SDEGADIS2 -->',//,3X,'BEGAN AT ',A40)
 4010   FORMAT(3X,'*** ELAPSED TIME *** ',1pG13.5,' min')
C
        STOP
        END
####
```

```
      function series(arg)
c
c*** This function estimates the infinite series used in the evaluation
c***  of the mass above a certain concentration level.
c
c**********************************************************************
c
c*** WARNING: This routine will overflow for arguments greater than 13.8
c
c**********************************************************************
c
      implicit real*8  (a-h,o-z), integer*4 (i-n)
c
      parameter (kmax= 100, error=1.d-4)
c
      common/alp/ alpha,alpha1
c
c*** initialize some variables
c
      pp = 1.D0/alpha1
      pprod = pp      ! for p*(p+1)*...*(p+k)
c
c*** calculate the first term
c
      coef = 2.D0/3.D0
      series = coef / pp * arg**1.5D0
      old = series
c
c
      do 200 kk=1,kmax
      pow = DBLE(kk) + 1.5D0
      plus = DBLE(kk+1)

      pprod = (pp + DBLE(kk)) * pprod
      coef = 2.D0 * plus**2 / DBLE(2*kk + 3)/ plus * coef

      series = series + coef / pprod * arg**pow

      denom = dmax1( (series+old), error)
      if( abs(series-old)/denom .le. error) return

      old = series
 200  continue
c
      write(6,900) arg
 900  format(' SERIES? Number of iterations too small for argument: ',
     $ 1pg13.5)
      call exit
      end
####
```

```
C......................................................
C
C       TIME SORT SUPERVISOR
C
        SUBROUTINE SORTS(TABLE)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

c
        include 'sys$degadis:DEGADIS3.dec/list'
C
        COMMON
     $/SORT/ TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
     $  Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $  Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $   TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $   TDISTO(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/SSCON/ NREC(maxnob,2),TO(maxnob),XV(maxnob)
     $/SORTIN/ TIM(maxnt),NTIM,ISTRT
     $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/CNOBS/ NOBS
C
        DIMENSION TABLE(1)
C
        REAL*8 ML,K
C
        CALL GETTIM
C
C
C*** TABLE(I) VALUES
C***    I               PARAMETER
C***    --              ---------
C***    1  11            DIST              1 TO 10 CURRENT READ
c***    2  12            Yc
C***    3  13            Cc                11 TO 20 PREVIOUS READ
c***    4  14            rho
c***    5  15            gamma
c***    6  16            temp
C***    7  17            SZ
C***    8  18            SY
C***    9  19            B
C***    10 20            TS
C
C***    21               DISTO
C***    22               INTERPOLATION FRACTION
C
        DO 100 I = 1,NOBS
C
        IT = 0
C
        DO 105 J=1,20
```

```
  105   TABLE(J) = 0.
C
        II = NREC(I,1)

        if( ii .eq. 0) goto 130
c
c*** read first record
c
        read(9,*) (table(k1),k1=1,9)
        table(10) = ts( t0(I), table(1) )
        table(21) = table(1)
c
c*** loop through and read each record even if not pertinent
c
        DO 110 J = 2,II
C
                DO K1 = 1,10
                KK = K1 + 10
                TABLE(KK) = TABLE(K1)
                enddo
C
        READ(9,*) (TABLE(K1),K1=1,9)
        TABLE(10) = TS( T0(I), TABLE(1) )
C
        itl = int( (table(10)-tim(1)) / (tim(2)-tim(1)) + 0.9999999 )
        itl = min( ntim, itl)
        itf = int( (table(20)-tim(1)) / (tim(2)-tim(1)) + 0.9999999 ) + 1
        itf = max( 1, itf)

        do it = itf, itl, 1             ! do all points in range
C
C*** RECORD AN INTERPOLATED TIME SORTED POINT.
C
        KSUB(IT) = KSUB(IT) + 1
C
        TABLE(22) = (TIM(IT) - TABLE(20))/(TABLE(10) - TABLE(20))
C
        TDISTO(I,IT) = TABLE(21)
        TDIST(I,IT)  = TABLE(11) + (TABLE(1) - TABLE(11)) * TABLE(22)
        Tyc(I,IT)    = TABLE(12) + (TABLE(2) - TABLE(12)) * TABLE(22)
        TCc(I,IT)    = TABLE(13) + (TABLE(3) - TABLE(13)) * TABLE(22)
        Trho(I,IT)   = TABLE(14) + (TABLE(4) - TABLE(14)) * TABLE(22)
        Tgamma(I,IT) = TABLE(15) + (TABLE(5) - TABLE(15)) * TABLE(22)
        Ttemp(I,IT)  = TABLE(16) + (TABLE(6) - TABLE(16)) * TABLE(22)
        TSZ(I,IT)    = TABLE(17) + (TABLE(7) - TABLE(17)) * TABLE(22)
        TSY(I,IT)    = TABLE(18) + (TABLE(8) - TABLE(18)) * TABLE(22)
        TB(I,IT)     = TABLE(19) + (TABLE(9) - TABLE(19)) * TABLE(22)
C
        enddo
  110   CONTINUE
C
```

```
C
  130   II = NREC(I,2)
        IF(II .EQ. 0) GO TO 100

        DO 200 J=1,II
C
                DO K1 = 1,10
                KK = K1 + 10
                TABLE(KK) = TABLE(K1)
                enddo
C
        READ(9,*) (TABLE(K1),K1=1,7)
C
        TABLE(8) = RT2*DELTA*(TABLE(1) + XV(I))**BETA
        TABLE(9) = 0.
        TABLE(10) = TS(TO(I),TABLE(1))
C
        itl = int( (table(10)-tim(1)) / (tim(2)-tim(1)) + 0.9999999 )
        itl = min( ntim, itl)
        itf = int( (table(20)-tim(1)) / (tim(2)-tim(1)) + 0.9999999 ) + 1
        itf = max( 1, itf)

        do it = itf, itl, 1             ! do all points in range
C
C*** RECORD A TIME SORTED VALUE
C
        KSUB(IT) = KSUB(IT) + 1
        TABLE(22) = (TIM(IT) - TABLE(20))/(TABLE(10) - TABLE(20))
C
        TDISTO(I,IT) = TABLE(21)
        TDIST(I,IT)  = TABLE(11) + (TABLE(1) - TABLE(11)) * TABLE(22)
        Tyc(I,IT)    = TABLE(12) + (TABLE(2) - TABLE(12)) * TABLE(22)
        TCc(I,IT)    = TABLE(13) + (TABLE(3) - TABLE(13)) * TABLE(22)
        Trho(I,IT)   = TABLE(14) + (TABLE(4) - TABLE(14)) * TABLE(22)
        Tgamma(I,IT) = TABLE(15) + (TABLE(5) - TABLE(15)) * TABLE(22)
        Ttemp(I,IT)  = TABLE(16) + (TABLE(6) - TABLE(16)) * TABLE(22)
        TSZ(I,IT)    = TABLE(17) + (TABLE(7) - TABLE(17)) * TABLE(22)
        TSY(I,IT)    = TABLE(18) + (TABLE(8) - TABLE(18)) * TABLE(22)
        TB(I,IT)     = TABLE(19) + (TABLE(9) - TABLE(19)) * TABLE(22)
C
        enddo
  200   CONTINUE
C
  100 CONTINUE
C
        CALL SORTS1(TABLE)
C
        RETURN
        END
####
```

```
C.............................................................
C
C
      SUBROUTINE SORTS1(TABLE)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C

      include 'sys$degadis:DEGADIS3.dec/list'
C
      COMMON
     $/SORT/ TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
     $  Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $  Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $  TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $  TDISTO(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/SSCON/ NREC(maxnob,2),TO(maxnob),XV(maxnob)
     $/SORTIN/ TIM(maxnt),NTIM,ISTRT
     $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/ALP/ ALPHA,alpha1
     $/CNOBS/NOBS
C
      DIMENSION TABLE(1)
C
      REAL*8 ML,K
      logical cflag
C
C*** DETERMINE IF ANY TIME VECTORS HAVE NO ENTRIES
C
      DO 192 I=1,NTIM
  192 IF(KSUB(I).GT. 2) GO TO 194         ! 1or2 points is of little value
      call trap(23)
  194 ISTRT = I
      DO 196 I=ISTRT,NTIM
  196 IF(KSUB(I).LE. 2) GO TO 198
      GO TO 199
  198 NTIM = I - 1
  199 CONTINUE
C
C*** REVERSE TIME SORTED VECTORS
C
C*** At this point, the results from Observer #1 is located in the
c***    first row of each vector.  But the information contributed by
c***    Observer #1 is the information which is the farthest (TDIST)
c***    from the source.  If the output is printed by time, the
```

```
c***   distances will decrease instead of increase.  So, we need to
c***   reverse each of the columns so that the downwind distances
c***   increase as you move down the column.  At the same time,
c***   ensure that all of the nonzero entries in a column are at
c***   the top of the column.
c
        DO 200 K1 = ISTRT,NTIM
c
        II = KSUB(K1)
        DO 170 J = 1,NOBS
c
c       IF(TDIST(J,K1).NE.0. .OR. TB(J,K1).NE.0.) GO TO 180
        IF( Tsz(J,K1).NE.0. ) GO TO 180
  170   CONTINUE
  180   II = II + J - 1
c
        DO 190 J = 1,II
c
        TABLE(II +        + 1 - J) =    TCc(J,K1)
        TABLE(II +   NOBS + 1 - J) =    Tyc(J,K1)
        TABLE(II + 2*NOBS + 1 - J) =   Trho(J,K1)
        TABLE(II + 3*NOBS + 1 - J) = Tgamma(J,K1)
        TABLE(II + 4*NOBS + 1 - J) =  Ttemp(J,K1)
        TABLE(II + 5*NOBS + 1 - J) =    TSY(J,K1)
        TABLE(II + 6*NOBS + 1 - J) =    TSZ(J,K1)
        TABLE(II + 7*NOBS + 1 - J) =     TB(J,K1)
        TABLE(II + 8*NOBS + 1 - J) = TDISTO(J,K1)
        TABLE(II + 9*NOBS + 1 - J) =  TDIST(J,K1)
c
  190   CONTINUE
c
        DO 210 J = 1,II
c
           TCc(J,K1) = TABLE(J)
           Tyc(J,K1) = TABLE(J +   NOBS)
          Trho(J,K1) = TABLE(J + 2*NOBS)
        Tgamma(J,K1) = TABLE(J + 3*NOBS)
         Ttemp(J,K1) = TABLE(J + 4*NOBS)
           TSY(J,K1) = TABLE(J + 5*NOBS)
           TSZ(J,K1) = TABLE(J + 6*NOBS)
            TB(J,K1) = TABLE(J + 7*NOBS)
        TDISTO(J,K1) = TABLE(J + 8*NOBS)
         TDIST(J,K1) = TABLE(J + 9*NOBS)
c
  210   CONTINUE
  200   CONTINUE
c
c
        if(sigx_flag.eq. 0.) then                    ! no correction
c
     ,      write(lunlog,*) ' No X-direction dispersion correction'
```

```
                DO 220 K1 = ISTRT,NTIM
                II = KSUB(K1)
                DO 220 I = 1,II
                TCcSTR(I,K1) = TCc(i,k1)
  220           CONTINUE
                goto 400
                endif
C
C*** GENERATE TCcSTR -- CENTER LINE CONCENTRATION CORRECTED FOR
C***  DOWNWIND DISPERSION.
C
        DO 230 K1 = ISTRT,NTIM
C
        II = KSUB(K1)
        DO 240 I = 1,II
C
c calculation for XP = TDIST(I,K1)
c
        TCcSTR(I,K1) = 0.
C
        DO 260 J = 1,II
C
        TABLE(J) = 0.
        DIST = TDIST(J,K1) - Tdist0(J,K1)
        deltax = ABS(tdist(i,k1) - tdist(j,k1))
c
        if(dist.lt. sigx_min_dist) then
                if(i.eq. j) then                ! i.e. deltax = 0.
                        table(j) = (tdist(j+1,k1)- tdist(j-1,k1))/2.
                        if(j.eq.1)table(j)= (tdist(2,k1)- tdist(1,k1))/2.
                        if(j.eq. ii)table(j)=tdist(ii,k1)- tdist(ii-1,k1)
                        table(j) = TCc(j,k1) / table(j) * RT2*SQRTPI
                endif
                goto 260
                endif
c
        SX = sigx_coeff* DIST**sigx_pow
C
        ARG = (deltax/SX)**2/2.
C
        IF(ARG .EQ. 0.) TABLE(J) = TCc(J,K1)/SX
        IF(ARG .NE. 0. .AND. ARG .LE. 80.)
     $                  TABLE(J) = TCc(J,K1)/SX/EXP(ARG)
C
  260   CONTINUE
C
        III = KSUB(K1)
        TCcSTR(I,K1) = TABLE(1)* (TDIST(2,K1)- TDIST(1,K1))/2.
        TCcSTR(I,K1) = TABLE(iii)* (TDIST(iii,K1)- TDIST(iii-1,K1))
        1                                               + TCcSTR(I,K1)
        iii = ksub(k1) - 1
```

```
C
        DO 280 J = 2,III
C
        TCcSTR(I,K1) = TABLE(J)* (TDIST(J+1,K1)- TDIST(J-1,K1))/2.
     1                                                 + TCcSTR(I,K1)
C
  280   CONTINUE
c
        TCcSTR(I,K1) = TCcSTR(I,K1)/RT2/SQRTPI
c
c*** correct yc, rho, and temp values
c
              cc = Tccstr(i,k1)
        if(isofl.eq. 1 .or. ihtfl.eq. 0) then
              call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enth,temp)
        else
              enth = Tgamma(i,K1)
              call adiabat(-1,wc,wa,yc,ya,cc,rho,wm,enth,temp)
        endif
        Tyc(i,K1) = yc
        Trho(i,K1)= rho
C
  240   CONTINUE
C
  230   CONTINUE
C
C
C*** Estimate the mass between the UFL and LFL
C
  400   continue
c
        cflag = isofl.eq.1 .or. ihtfl.eq.0
        if(cflag) then
           call adiabat(2,aa,dd,gas_ufl,ee,chi ,gg,hh,pp,oo)
           call adiabat(2,aa,dd,gas_lfl,ee,clow,gg,hh,pp,oo)
        endif
c
        DO 430 K1 = ISTRT,NTIM
        kk  = k1+2*maxnob
        kk1 = k1+3*maxnob
        table(kk) = 0.
        table(kk1)= 0.
C
        II = KSUB(K1)
        DO 460 J = 1,II ! evaluate the function at each point in space
C
C  initialize some values
c
        TABLE(J) = 0.
        jj = j+maxnob
        TABLE(JJ) = 0.
        cc = TCcstr(j,k1)
        bb = tb(j,k1)
        sy = tsy(j,k1)
        sz = tsz(j,k1)
        gamma = tgamma(j,k1)

        if(.not.cflag) then
           call adiabat(-2,aa,dd,gas_ufl,ee,chi ,gg,hh,gamma,oo)
           call adiabat(-2,aa,dd,gas_lfl,ee,clow,gg,hh,gamma,oo)
        endif
c
c*** Calculate the derivative for the total mass above the UFL and LFL
c
              gamhi  = 2.D0* Cc * Bb * Sz / alpha1
       gammax = 2.D0 *Cc *Sz *GAMMAF /alpha1 *(Bb +sqrtpi/2.D0 *Sy)

         if(cc.gt.clow) then
              wlow = Dlog(cc/clow)
              gamlow = gaminc(1.D0/alpha1, wlow ) * gamhi
              table(j)= gamlow + 2.D0*clow*Sy*Sz/alpha1*series(wlow)
              table(j)= DMIN1( table(j), gammax )
         endif

         if(cc.gt.chi) then
              whi  = Dlog(cc/chi )
              gamhi  = gaminc(1.D0/alpha1, whi  ) * gamhi
              table(jj) = gamhi+ 2.D0*chi *Sy*Sz/alpha1*series(whi)
              table(jj)= DMIN1( table(jj), gammax )
         endif
c
 460    continue
c
c*** now, finish the integration
c
        DO 450 J = 2,II          ! integrate in space at one value of time
        xx = (tdist(j,k1) - tdist(j-1,k1))
        arglow = (table(j) + table(j-1))/2.
        arghi  = (table(j+maxnob) + table(j-1+maxnob))/2.
        table(kk)  = arghi *xx + table(kk)
        table(kk1) = arglow*xx + table(kk1)
 450    continue
c
 430    continue
C
        RETURN
        END
####
```

```
C       SOURCE EQUATIONS -- Gas Blanket present

        SUBROUTINE SRC1(time,Y,D,PRMT)


        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADIS1.dec'

        parameter(      delt= 0.1D0,
        1               delto2= delt/2.D0,
        2               zero= 1.D-10,
        3               rcrit= 0.002D0)

         COMMON
      $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
      $    PFRACV(igen), PENTH(igen), PRHO(igen)
      $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtUH,XLI,
      $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
      $ htcut,ERNOBL,NOBLpt,crfger,epsilon
      $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
      $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
      $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
      $ humsrc
      $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
      $/vucom/ vua,vub,vue,vud,vudelta,vuflag
      $/com_enthal/ h_masrte,h_airrte,h_watrte
      $/ALP/ ALPHA,alpha1
      $/phicom/ iphifl,dellay
      $/sprd_con/ ce, delrhomin

        LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
        logical vuflag

        REAL*8 ML,K
        REAL*8 L,masrte,mole
        INTEGER R,mass,massc,massa,ebal,mbal
        DIMENSION Y(7),D(7),PRMT(25)
        DATA R/1/,mass/2/,massc/3/,massa/4/,ebal/5/,mbal/6/


        if( prmt(20).lt. 0.D0) vuflag = .false.

        if(Y(mass) .le. 0.D0) then
                wc = dmax1(prmt(15),1.d-10)
                if(wc.gt. 1.) wc=1.d-10
                wa = 1.D0 - wc
                enthalpy = wc*h_masrte               ! air contributes nothing
        else
                wc = Y(massc)/Y(mass)
                wa = Y(massa)/Y(mass)
```

```
            enthalpy = Y(ebal)/Y(mass)
      endif

      humsrc = (1.D0 - wc - wa*(1.D0+humid))/wc
      call tprop(1,wc,wa,enthalpy,yc,ya,mole,temp,rho,cp)

      RADP = AFGEN2(PTIME,R1T,TIME,'R1TSRC')
      hei   = dmax1( Y(mass)/pi/Y(r)/Y(r)/rho , 0.0D0 )
      delrho = rho-rhoa
      if(delrho .lt. 0.D0) delrho = 0.D0
      gprime = g*delrho/rhoa *hei

C*** CALCULATE D(R),airrte,vel

      D(R) = 0.D0
      vel = 0.D0
      airrte = 0.D0
      Ri = 0.D0
      D(mbal)  = 0.D0

      IF(Gprime.GT. 0.D0) then
         slump = Ce*sqrt(Gprime)

         if(vuflag) then                  ! momentum balance
              iii = 0          ! initialize loop counter
              vel = prmt(14)            ! old velocity value
              velmin = 0.D0
              velmax= dmax1( slump, 0.1D0, vel)

100           hh = vel*vel/Ce/Ce/g/ (delrho/rhoa)
              rh = Y(r)-vua*vub*hh
              value = Y(r)**2/rh**2

                 if(prmt(25).ge. prmt(24)) then       ! hh .ge. ht

              ht = 2.D0*(value*hei - vua*hh*(value-1.D0)) - hh
              velc = Y(mbal)/(0.4D0*pi*rho*(2.D0/3.D0*ht + hh)*rh**3/Y(r)
     1           + 2.D0/3.D0*pi*vua*rho*hh* (Y(r)**2 - rh*rh*rh/Y(r))
     2           + vue*pi*Y(r)*hei**2*rhoa)
              D(mbal) = pi*g*delrho*Y(r)*ht**2
     1           - vua*vud*pi*rhoa*Y(r)*hh*vel**2
                 else

              ht = value*hei - vua*hh*(value-1.D0)
              velc = Y(mbal)/(2.D0/3.D0*pi*rho*ht*rh**3/Y(r)
     1           + 2.D0/3.D0*pi*vua*rho*hh* (Y(r)**2 - rh*rh*rh/Y(r))
     2           + vue*pi*Y(r)*hei**2*rhoa)
              D(mbal) = pi*g*delrho*(rh*ht**2 + vua*vub*hh*hh*hh)
     1           - vua*vud*pi*rhoa*Y(r)*hh*vel**2
                 endif
                 dif = abs(vel-velc)             ! convergence check
                 sum = abs(vel) + abs(velc) + zero

                    if(dif/sum .le. rcrit) then
                         vel = (vel+velc)/2.D0
                         prmt(13) = vel

                             if(vel .gt. 0.D0) then
                                 Ri = gprime / vel**2
                airrte= 2.*pi* epsilon/Ri *rhoa*Y(r)*hei* vel
                                 D(r) = vel
                                 prmt(20) = slump
                             endif
                    else

                        dif = vel-velc

                        if(velc.lt.velmin) velmin= dmax1(velc, 0.D0)
                        if(velc.gt.velmax) velmax=velc

                        if(dif .gt. 0.D0) then
                            velmax = vel
c                           vel = 0.5D0*(velmax-velmin) + velmin
                            vel = 0.382D0*(velmax-velmin) + velmin
                        else
                            velmin = velc
c                           vel = (1.D0-0.5D0)*(velmax-velmin) + velmin
                    vel = (1.D0-0.382D0)*(velmax-velmin) + velmin
                        endif

                        iii = iii+1
                        if(iii .gt. 40) then
                            vel = min(velmin,velmax)
                            if(vel .gt. slump) then
                                vel = slump
                                prmt(13) = vel
                                if(vel .gt. 0.D0) then
                                    Ri = gprime / vel**2
                                    airrte= 2.*pi* epsilon/Ri
                                              *rhoa*Y(r)*hei* vel
                                    D(r) = vel
                                    prmt(20) = slump
                                endif
                                goto 200
                            endif
                            write(6,*) 'Time, vel, slump: ',time,vel,slump
                            write(6,*) 'prmt(14), velc: ',prmt(14),velc
                            stop'SRC1 velocity loop'
                      endif
                      goto 100
                  endif
```

```
          else

              vel= slump      ! gravity slumping
              hh = hei
              ht = hei
              Ri = gprime / vel**2
              airrte= 2.D0*pi* epsilon/Ri *rhoa*Y(r)*hei* vel
              D(r) = vel
          endif
      endif
c
  200 IF(delrho.Lt.delrhomin .and. u0.ne.0.) then
              D(R) = 0. ! not for windless cases
              airrte = 0.
              endif
c
      area = pi * (Y(r)**2 - radp**2)
      IF(Y(R).Le. RADP) THEN
              AREA = 0.D0
              Y(R) = RADP + zero
              IF(time.gt. delto2) then ! delt; num prob
                      D(R) = dmax1(0.D0,
     1                        ((AFGEN2(PTIME,R1T,TIME+delto2,'R1TSRD')-
     2                AFGEN2(PTIME,R1T, (TIME-delto2),'R1TSRe'))/ delt))
              else
                      D(R) = dmax1(0.D0,
     1                        ((AFGEN2(PTIME,R1T,delto2,'R1TSRD')-
     2                        AFGEN2(PTIME,R1T, 0.D0,'R1TSRe'))/ delto2))
              endif
      ENDIF
c
c     calculate totrteout
c
      masrte = AFGEN2(PTIME,ET,TIME,'src1')
      PWCP = AFGEN2(PTIME,PWC,TIME,'src1')
      TOTPRT = MASRTE/PWCP
      HPRIM = AFGEN2(PTIME,PENTH,TIME,'src1')
      L = 2.0D0 * Y(R)
c
      cc = wc*rho
c
c
      qstrmx = 0.D0
      if(u0 .ne. 0.D0)
     1   qstrmx = cc*K*USTAR*ALPHA1*dellay/(dellay-1.D0)/phihat(rho,L)
c
c
c
      qstril = qstrmx * pi*L*L/4.D0
      totrteout = qstrll/wc
c
c     surface effects
c
      watrte = 0.D0
      surface_q = 0.D0
      yw = 1.D0-ya-yc
      yw = min( max(yw, 0.0D0), 1.0D0)
      call surface(temp,hei,rho,mole,cp,yw,watrte,surface_q)
      surface_q = area * surface_q
      if(surface_q.lt. 0.D0) surface_q = 0.D0 ! don't let the cloud cool
      watrte = area * watrte
c
 500  totrtein  = airrte + TOTPRT + watrte
c
      IF(totrtein.lt.totrteout .and. .not.check2
     1          ) then  ! check2 is True for HSE type spills
            D(R) = 0.
            if(hei.gt.srccut .and. Y(r).gt.srccut) then
            dHdt = (totrtein - totrteout)/3.D0/pi/ Y(r)/Y(r)/rho
                  D(R) = Y(R)/Hei * dHdt
c                       ! Let cloud radius shrink when...
            endif       ! cloud height is decreasing unless...
      if( Y(R).le.0.01D0*rmax .and. masrte.eq.0.D0)
     $      D(R) = 0. ! the primary source has stopped. tos;6mar86
      if( Y(R).le.hei .and. masrte.eq.0.D0)
     $      D(R) = 0. ! the cloud height > the cloud radius. tos;16nov87
            endif
c
c     CALCULATE D(mass),D(massc),D(massa),D(anything left)
c
      D(mass)  = totrtein - totrteout
      D(massc) = masrte - qstrll
      D(massa) = (airrte + MASRTE*(1.D0/PWCP - 1.D0))/(1.D0+humid)
     $                             - wa/wc*qstrll
      D(ebal)  = 0.D0
      if(ihtfl.ne. 0) ! equivalent to adiabatic mixing from TPROP for ihtfl=0
     $      D(ebal)  = HPRIM*TOTPRT + h_airrte*airrte
     $            + h_watrte*watrte - enthalpy*totrteout + surface_q
c
c
            uheff = qstrmx*L/cc
      sz = 0.D0
      if(u0 .ne. 0.D0) sz = ( uheff*alpha1/u0/z0 )**(1.D0/alpha1) * z0
c
c
C
      PRMT(6) = QSTRMX
      prmt(7) = sz
      prmt(8) = hei
      prmt(9) = rho
      prmt(10)= Ri
      prmt(11)= yc
      prmt(12)= ya
      prmt(13)= D(r)
      prmt(16) = wc
      prmt(17) = wa
      prmt(18) = enthalpy
      prmt(19) = temp
      prmt(21) = masrte
      prmt(22) = ht
      prmt(23) = hh
      RETURN
      END
c
c
C.............................................................
C
C       SUBROUTINE FOR OUTPUT FROM SOURCE in the presence of a Blanket
C
        SUBROUTINE SRC1O(TIME,Y,DERY,IHLF,NDIM,PRMT)

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        include 'sys$degadis:DEGADIS1.dec'
C
        COMMON
     $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl,swcl,swal,senl,srhl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/ALP/ ALPHA,alpha1
C
        LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
        DIMENSION Y(6),DERY(6),PRMT(25)
        DIMENSION CURNT(iout_src),BKSP(iout_src),OUTP(iout_src)
C
        DATA I/0/,III/0/
        DATA EMAX/0./,tlast/0./
c
        REAL*8 ML,K
        INTEGER R,mass,massc,massa,ebal
        DATA R/1/,mass/2/,massc/3/,massa/4/,ebal/5/
c
        data nrec1/0/
C
        I = I + 1
      III = III + 1
C
      qstr = prmt(6)
      sz = prmt(7)
      hei = prmt(8)
      rho = prmt(9)
      Ri  = prmt(10)
      yc = prmt(11)
      ya = prmt(12)
      vel = prmt(13)
      prmt(14) = vel
      if(vel .gt. prmt(20)) prmt(20) = -prmt(20)
      prmt(15) = prmt(16)     ! wc
      wc = prmt(16)
      cc = wc * rho
      wa = prmt(17)
      enthalpy = prmt(18)
      temp = prmt(19)
      prmt(24) = prmt(22)     ! ht
      prmt(25) = prmt(23)     ! hh
c
      IF(hei .Le. 0.000) GO TO 1000
C
      QSAV = PI*Y(R)*Y(R)*qstr
      IF(QSAV .LT. EMAX) GO TO 110
      EMAX = QSAV
      RM = Y(R)
      SZM = SZ
 110  CONTINUE
      RMAX = dMAX1(RMAX,Y(R))
C
      IF(hei .Le. srccut) GO TO 1000
      if(cc .le. cclow .and. u0 .eq. 0.) goto 1000   ! no wind
      if(time.gt.tend+1. .and. u0.eq.0. .and. vel.eq.0.)goto 1000!no wind LNG
C
      IF(I .NE. 1) GO TO 115
      CURNT(1) = TIME
      CURNT(2) = Y(R)
      CURNT(3) = hei
      CURNT(4) = qstr
      CURNT(5) = sz
      CURNT(6) = yc
      CURNT(7) = ya
      CURNT(8) = rho
      CURNT(9) = ri
      CURNT(10)= wc
      CURNT(11)= wa
      CURNT(12)= enthalpy
      CURNT(13)= temp
      III = 1
      GO TO 125
```

```
  115 IF(I .EQ. 0) RETURN
C
      DO 116 II=1,iout_src
  116 BKSP(II) = CURNT(II)
C
      CURNT(1) = TIME
      CURNT(2) = Y(R)
      CURNT(3) = hei
      CURNT(4) = qstr
      CURNT(5) = sz
      CURNT(6) = yc
      CURNT(7) = ya
      CURNT(8) = rho
      CURNT(9) = ri
      CURNT(10)= wc
      CURNT(11)= wa
      CURNT(12)= enthalpy
      CURNT(13)= temp
C
      ERM = 0.
      ermss = 0.
      DO 120 II=2,iout_src
      div = curnt(ii)
      if(div .eq. 0.) div = srcoer
      ER1 = ABS( (CURNT(II)-BKSP(II))/div )
      ER2 = ABS( (CURNT(II)-OUTP(II))/div )
      if(II.ne.3 .and. ii.ne.9 .and. ii.ne.12 .and. ii.ne.7 .and. ii.ne.11)
     1       ermss = dMAX1(ER1,ER2,ERMss) ! ex hei,QSTR,Ri,enth,wa,ya for SS
  120 ERM = dMAX1(ER1,ER2,ERM)
C
      if(check4) then                   ! steady state
            if( .not. (vel.eq. 0..and.time.gt.srcss)) goto 124
  122             check3 = .true.
                  outcc = wc * rho
                  swcl = wc
                  swal = wa
                  srhl = rho
                  senl = enthalpy
                  outl  = 2.0D0 * Y(r)
                  Qstar = prmt(21)/pi/Y(r)**2
      if(u0.ne. 0.) sz= (alpha1/u0/z0*Qstar*outl/outcc)**(1.D0/alpha1)* z0
                  outsz = sz
                  outb  = pi*Y(r)**2 /outl/2.D0
                  goto 1000

  124       if(ermss .gt. srcoer) goto 125

            if( time-tlast .gt. srcss) goto 122
      return
            endif
c
```

```
      IF(ERM .LT. SRCOER) RETURN
C
  125 CONTINUE
      tlast = time
      DO 130 II=1,iout_src
      IF(III.EQ.1) BKSP(II) = CURNT(II)
  130 OUTP(II) = BKSP(II)
C
      III = 0
      NREC1 = NREC1 + 1
      WRITE(9,2000) (OUTP(II),II=1,iout_src)
      RETURN
C
 1000 CONTINUE
      I = -1
      IF(TIME .GE. TEND) CHECK3 = .TRUE.
      NREC1 = NREC1 + 1
      WRITE(lunlog,1100)
      WRITE(lunlog,*) Hei,TIME
      TSC1 = TIME
      if(hei .le. 0.) then
              hei = 0.
              y(r) = dmin1(rmax,y(r))
              endif
      WRITE(9,2000)
     1        TIME,Y(R),hei,qstr,sz,yc,ya,rho,ri,wc,wa,enthalpy,temp
      WRITE(lunlog,1110) NREC1
C
      PRMT(5) = 1.
C
      RETURN
 1100 FORMAT(5X,'VALUE OF Hei AT SOURCE TERMINATION -- @ TIME')
 1110 FORMAT(5X,'NUMBER OF LINES -->  ',I8)
 2000 format(1pg16.9,1x,1pg16.9,<iout_src-2>(1x,1pg13.6))
      END
####
```

```
      SUBROUTINE SRTOUT(OPNRUP, table)
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS3.dec/list'
C
      COMMON /SORT/TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
     $  Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $  Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $  TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $  TDISTO(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/SORTIN/TIM(maxnt),NTIM,ISTRT
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,temb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/alp/ alpha,alpha1
     ./oomsin/ oodist,avtime
C
      dimension table(1)
c
      logical cflag,cflag1
c
      character*3 gas_name
      character*40 OPNRUP
C
      OPEN(UNIT=8,TYPE='NEW',NAME=OPNRUP,CARRIAGECONTROL='FORTRAN')
C
      WRITE(8,1100)
      if(sigx_flag.eq. 0.) then
              write(8,1102)
      else
              write(8,1104)
              write(8,1105) sigx_coeff,sigx_pow,sigx_min_dist
              endif
c
      cflag = isofl.eq. 1.or. ihtfl.eq. 0
      cflag1= isofl.eq.1
      if(cflag) then
              call adiabat(2,wc,wa,gas_lfl,ya,cc_lfl,r,w,t,tt)
              call adiabat(2,wc,wa,gas_ufl,ya,cc_ufl,r,w,t,tt)
              endif
C
      DO 110 I=ISTRT,NTIM
C
      WRITE(8,1119)
      WRITE(8,1119)
      WRITE(8,1110) TIM(I)
      if(cflag1) then
```

```
              WRITE(8,1116) gas_zsp,(100.*gas_lfl),(100.*gas_ufl)
              WRITE(8,1118)
      else
              WRITE(8,1115) gas_zsp,(100.*gas_lfl),(100.*gas_ufl)
              WRITE(8,1117)
              endif
      WRITE(8,1119)
      ip = 0
      II = KSUB(I)
c
      DO 120 J=1,II
c
      dist  = tdist(j,i) + oodist
      cc    = tccstr(j,i)
      rho   = Trho(j,i)
      yc    = Tyc(j,i)
      temp  = Ttemp(j,i)
      gamma = Tgamma(j,i)
      b     = tb(j,i)
      sz    = tsz(j,i)
      sy    = tsy(j,i)
      blfl  = 0.
      bufl  = 0.
c
      if(.not.cflag) then
              call adiabat(-2,wc,wa,gas_lfl,ya,cc_lfl,r,w,gamma,tt)
              call adiabat(-2,wc,wa,gas_ufl,ya,cc_ufl,r,w,gamma,tt)
              endif
c
      arg = (gas_zsp/sz)**alpha1
      if(arg .ge. 80.) goto 600
c
      ccz = cc/exp(arg)
      if(ccz .lt. cc_lfl) then
              if(cflag1) then
      WRITE(8,1120) DIST,yc,Cc,rho,temp,B,SZ,SY
                      else
      WRITE(8,1120) DIST,yc,Cc,rho,gamma,temp,B,SZ,SY
                      endif
              goto 600
              endif
      arg = -(dlog(cc_lfl/cc) + (gas_zsp/sz)**alpha1)
      blfl = sqrt(arg)*sy + b
c
      if(ccz .lt. cc_ufl) then
              if(cflag1) then
      WRITE(8,1120) DIST,yc,Cc,rho,temp,B,SZ,SY,blfl
                      else
      WRITE(8,1120) DIST,yc,Cc,rho,gamma,temp,B,SZ,SY,blfl
                      endif
              goto 600
```

```
                endif
        arg = -(dlog(cc_ufl/cc) + (gas_zsp/sz)**alpha1)
        bufl = sqrt(arg)*sy + b
             if(cflag1) then
        WRITE(8,1120) DIST,yc,Cc,rho,temp,B,SZ,SY,blfl,bufl
                     else
        WRITE(8,1120) DIST,yc,Cc,rho,gamma,temp,B,SZ,SY,blfl,bufl
                     endif
c
  600   continue
        ip = ip + 1
        if(ip .eq. 3) then
               ip = 0
               write(8,1119)
               endif
  120   CONTINUE
c
c*** summarize the mass above the UFL and LFL
c
        aufl = table(i+2*maxnob)
        alfl = table(i+3*maxnob)
        write(8,8000) 100.*gas_ufl,100.*gas_lfl,alfl-aufl,alfl
 8000   format(//,' For the UFL of ',1pg13.5,' mole percent, and',
     $' the LFL of ',1pg13.5,' mole percent:',
     $//,' The mass of contaminant between the UFL and LFL is:'
     $,1pg13.5,' kg.',/,' The mass of contaminant above the LFL is: ',
     $1pg13.5,' kg.')
  110   CONTINUE
C
        CLOSE(UNIT=8)
C
C
 1100   FORMAT(1H0,5X,'Sorted values for each specified time.')
 1102   format(1H0,5x,'X-Direction correction was NOT applied.')
 1104   format(1H0,5x,'X-Direction correction was applied.')
 1105   format(1h ,5x,5x,'Coefficient:       ',1pg13.5,/,
        1       1h ,5x,5x,'Power:             ',1pg13.5,/,
        1       1h ,5x,5x,'Minimum Distance: ',1pg13.5' m')
 1110   FORMAT(1H0,5X,'Time after beginning of spill ',G14.7,' sec')
 1115   FORMAT(1H0,1X,'Distance',2x,3x,'Mole',3x,
        1       'Concentration',1x,'Density',2x,3x,'Gamma',3x,
        1       'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
        1       'Width at z=',0pf6.2,' m to:',/,1x,11x,1x'Fraction',2x,
        1       11x,11x,11x,11x,3x,'Width',3x,11x,9x,
        1       2(1pg9.3,'moleX',1x))
 1116   FORMAT(1H0,1X,'Distance',2x,3x,'Mole',3x,
        1       'Concentration',1x,'Density',2x,
        1       'Temperature',3x,'Half',4x,4x,'Sz',5x,4x,'Sy',5x,
        1       'Width at z=',0pf6.2,' m to:',/,1x,11x,1x'Fraction',2x,
        1       11x,11x,11x,3x,'Width',3x,11x,9x,
        1       2(1pg9.3,'moleX',1x))
```

```
1117    FORMAT(1H ,4X,'(m)',4x,11x,
        1       2(1X,'(kg/m**3)',1x),11x,4x,'(K)',
        1       5(8X,'(m)'))
1118    FORMAT(1H ,4X,'(m)',4x,11x,
        1       2(1X,'(kg/m**3)',1x),4x,'(K)',
        1       5(8X,'(m)'))
1119    FORMAT(1H )
1120    FORMAT(1H ,3(1X,1PG9.3,1X),2x,0pf7.4,2x,1X,1PG10.3,1X,
        1       6(1X,1PG9.3,1X))
C
        RETURN
        END
####
```

```
C.............................................................
C
C
      SUBROUTINE SSG(DIST,Y,Dery,PRMT)

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
      DIMENSION Y(1),Dery(1),PRMT(1)
C
      include 'sys$degadis:DEGADIS2.dec'
c
      parameter (zero=1.D-10, rcrit=2.5D-3)
c
      COMMON
     $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
c
      REAL*8 K,ML

      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      INTEGER rhouh,dh, mhi, mlow
      DATA rhouh/1/,dh/2/, mhi/3/, mlow/4/
C
C*** PRMT(I) I/O
C***    I             VALUE          IN/OUT
C***    --            -----          ------
C***    6               E              IN
C***    7               Cc             OUT
C***    8               XV(I)          IN
C***    9               TO(I)          IN
C***    10              -              -
C***    11              NREC(I,2)      OUT -- STARTS OUTPUT UNIT=9
C***    12              DIST           OUT
c***    13              sz
c***    14              yc             out
c***    15              rho            out
c***    16              temp           out
c***    17              gamma          out
c***    18
c***    19
c***    20              rholay
c***    21              sz
C
```

```
      XVl = PRMT(8)
      SY = RT2*DELTA*(DIST + XVl)**BETA
      Erate = PRMT(6)
c
c
      sz0 = prmt(22)
      sz = sz0
C
C*** MATERIAL BALANCE
C
      iii = 0
 100  Cc = Erate*ALPHA1*(Z0/SZ)**ALPHA/U0/SZ/SQRTPI/SY
c
      cclay = cc/dellay
      call addheat(cclay,Y(dh),rholay,temlay,cp)
      prod = dmax1( Y(rhouh)/rholay/prmt(18), zero)
      sz = ( prod ) **(1./alpha1) * z0
      dif = abs(sz - sz0)/(abs(sz)+abs(sz0)+zero)
      if(dif.gt. rcrit) then
            iii = iii+1
            if(iii.gt. 20) call trap(33)
            sz0 = sz
            goto 100
            endif
      prmt(20) = rholay
      prmt(21) = sz
      HEFF = GAMMAF*SZ/ALPHA1

      call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enth,temp)
      call adiabat(0,wc,wa,yclay,ya,cclay,rholam,wml,enth,temlam)
      rit = 0.
c
      if(isofl.eq.0 .and. ihtfl.ne.0) then
            call addheat(cc,dellay*Y(dh),rho,temp,cp)
            rit = rift(temp,heff)
            endif
C
      RISTR = RIF(RHO,HEFF)
      PHI = PHIF(RISTR,rit)
C
      dery(rhouh) = prmt(19)/phi
      heigh = heff*dellay
      yw = 1.-yclay-ya
      yw = min( max( yw, 0.000), 1.000)
      call surface(temlay,heigh,rholay,wml,cp,yw,watrte,qrte)
      if(temp.ge. tsurf .or. temlay.ge. tamb) qrte = 0.
      dery(dh) = ( qrte/dellay-Y(dh)*Dery(rhouh) )/Y(rhouh)
C
c
c*** Calculate the derivative for the total mass above the UFL and LFL
c
```

```
        gamma = (rho-rhoa)/cc                                   ! gamma

        if(check4) then
          DERY(mlow) = 0.
          DERY(mhi)  = 0.
           if( isofl.eq.1 .or. ihtfl.eq.0 ) then
                call adiabat(2,aa,dd,gas_ufl,ee,chi ,gg,hh,pp,oo)
                call adiabat(2,aa,dd,gas_lfl,ee,clow,gg,hh,pp,oo)
          else
                call adiabat(-2,aa,dd,gas_ufl,ee,chi ,gg,hh,gamma,oo)
                call adiabat(-2,aa,dd,gas_lfl,ee,clow,gg,hh,gamma,oo)
          endif

       gammax = sqrtpi * Cc * Sz * Sy * GAMMAF / alpha1

         if(cc.gt.clow) then
               wlow = Dlog(cc/clow)
               DERY(mlow)= 2.D0*clow*Sy*Sz/alpha1*series(wlow)
               DERY(mlow)= DMIN1( DERY(mlow), gammax )
        endif

        if(cc.gt.chi ) then
               whi  = Dlog(cc/chi )
               DERY(mhi) = 2.D0*chi *Sy*Sz/alpha1*series(whi )
               DERY(mhi) = DMIN1( DERY(mhi ), gammax )
        endif
      endif
c
      PRMT(7) = Cc
      PRMT(12) = DIST
      prmt(14) = yc
      prmt(15) = rho
      prmt(16) = temp
      prmt(17) = gamma
      RETURN
      END
####
```

```
C..............................................................
C
      SUBROUTINE SSGOUT(X,Y,D,INLF,NDIM,PRNT)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS2.dec'
c
      parameter (nssg=7, zero=1.e-10)
c
      DIMENSION Y(1),D(1),PRNT(1),BKSP(nssg),OUT(nssg),CURNT(nssg)
C
      COMMON
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/STP/ STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/STOPIT/ TSTOP
c
      REAL*8 K,ML
C
C*** PARAMETER OUTPUT
C
C***  FROM SSG             OUTPUT TO MODEL
C***  --------             --------------
C***    X                        DIST
C***    PRNT(7)                  Cc
C***    Y(1)                     SZ
c***    prmt(14)                 yc
c***    prmt(15)                 rho
C***    prmt(16)                 temp
c***    prmt(17)                 gamma
C
      ERM = 0.
      TOl = PRNT(9)
      TSL = TS(TOl,X)
      prmt(22) = prmt(21)     ! sz
C
      IF(PRNT(11) .NE. 0.) GO TO 90
C
C*** STARTUP FOR OUTPUT ROUTINE
C
      RII = -100./STPG
      RI = 0.
      CURNT(1) = X
      curnt(2) = prmt(14)     ! yc
      CURNT(3) = PRNT(7)      ! cc
      curnt(4) = prmt(15)     ! rho
      curnt(5) = prmt(17)     ! gamma
      curnt(6) = prmt(16)     ! temp
      CURNT(7) = prmt(21)      ! sz
C
   90 CONTINUE
C
C*** RECORD THE CURRENT AND PREVIOUS RECORDS
C
      RI = RI + 1.
C
      DO 100 II=1,nssg
  100 bksp(II) = curnt(II)
c
      CURNT(1) = X
      curnt(2) = prmt(14)      ! yc
      CURNT(3) = PRMT(7)       ! cc
      curnt(4) = prmt(15)      ! rho
      curnt(5) = prmt(17)      ! gamma
      curnt(6) = prmt(16)      ! temp
      CURNT(7) = prmt(21)      ! sz
C
c*** stop integration when cc<cclow and time is satisfied
C
      IF(PRMT(7).LT.CcLOW .AND. TSL.GE.TSTOP) GO TO 1000
C
C*** CHECK FOR OUTPUT
C
      DO 110 II=3,nssg
      ER1 = ABS( (CURNT(II)-BKSP(II))/(CURNT(II)+zero) )
      ER2 = ABS( (CURNT(II)-OUT(II))/(CURNT(II)+zero) )
  110 ERM = dMAX1(ER1,ER2,ERM)
C
C*** OUTPUT RECORD IF ODLG IS EXCEEDED OR 100 METERS SINCE LAST OUTPUT
C
      DX = CURNT(1) - OUT(1)
      IF( RI.NE.1. .AND. ERM.LT.ODLG .AND. DX.LE.ODLLG) RETURN
C
C*** RECORD THE LAST POINT TO BE UNDER THE ERROR CRITERIA. IN CASE
C*** THE FIRST POINT AFTER A RECORD EXCEEDS THE ERROR BOUND, RECORD
C*** THAT POINT AS WELL.
C
      DO 120 II=1,nssg
      IF(RI .EQ. RII+1.) BKSP(II) = CURNT(II)
  120 OUT(II) = BKSP(II)
C
      RI = RII
      PRMT(11) = PRMT(11) + 1.
C
      WRITE(9,*) (OUT(II),II=1,nssg)
      RETURN
C
 1000 CONTINUE
C
```

```
C*** STOP INTEGRATION
C
      PRMT(12) = X
      TSTOP = TSL
      PRMT(11) = PRMT(11) + 1.
      WRITE(9,*) (CURNT(II),II=1,nssg)
C
      PRMT(5) = 1.
C
      RETURN
      END
####
```

```
C.............................................................
C
      SUBROUTINE SSGOUT(X,Y,D,IHLF,NDIM,PRMT)
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      parameter (nssg=9, zero=1.e-10)
C
      DIMENSION Y(1),D(1),PRMT(1),BKSP(nssg),OUT(nssg),CURNT(nssg)
C
      include 'sys$degadis:DEGADIS2.dec/list'
c
      COMMON
     $/PARM/U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/STP/STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/ALP/ALPHA,alpha1
C
c
      REAL*8 ML,K
C
C*** PARAMETER OUTPUT
C
C***  FROM SSG             OUTPUT TO MODEL
C***  --------             ---------------
C***    X                      DIST
C***    PRMT(7)                Cc
C***    Y(1)                   SZ
C***    PRMT(8)                XV
C
      ERM = 0.
      prmt(22) = prmt(21)
C
      IF(PRMT(11) .NE. 0.) GO TO 90
C
C*** STARTUP FOR OUTPUT ROUTINE
C
      RII = -100./STPG
      RI = 0.
      CURNT(1) = X
      CURNT(2) = PRMT(14)     ! yc
      CURNT(3) = prmt(7)      ! cc
      CURNT(4) = PRMT(15)     ! rho
      curnt(5) = prmt(17)     ! gamma
      curnt(6) = prmt(16)     ! temp
      curnt(7) = 0.0          ! b
      curnt(8) = prmt(21)     ! sz
      curnt(9) = rt2*delta*(x+prmt(8))**beta  ! sy
C
   90 CONTINUE
C
```

```
C*** RECORD THE CURRENT AND PREVIOUS RECORDS
C
       RI = RI + 1.
C
       DO 100 II=1,nssg
  100  BKSP(II) = CURNT(II)
       CURNT(1) = X
       CURNT(2) = PRMT(14)     ! yc
       CURNT(3) = prmt(7)      ! cc
       CURNT(4) = PRMT(15)     ! rho
       curnt(5) = prmt(17)     ! gamma
       curnt(6) = prmt(16)     ! temp
       curnt(7) = 0.0          ! b
       curnt(8) = prmt(21)     ! sz
       curnt(9) = rt2*delta*(x+prmt(8))**beta  ! sy
C
C*** STOP INTEGRATION WHEN Cc < CcLOW
C
       IF(PRMT(7).LT.CcLOW) GO TO 1000
C
C*** CHECK FOR OUTPUT
C
       DO 110 II=2,nssg
       if(ii.eq.7) goto 110
       ER1 = ABS( (CURNT(II)-BKSP(II))/(CURNT(II)+zero) )
       ER2 = ABS( (CURNT(II)-OUT(II))/(CURNT(II)+zero) )
  110  ERM = dMAX1(ER1,ER2,ERM)
C
C*** OUTPUT RECORD IF ODLG IS EXCEEDED OR 100 METERS SINCE LAST OUTPUT
C
       DX = CURNT(1) - OUT(1)
       IF( RI.NE.1. .AND. ERM.LT.ODLG .AND. DX.LE.ODLLG) RETURN
C
C*** RECORD THE LAST POINT TO BE UNDER THE ERROR CRITERIA. IN CASE
C*** THE FIRST POINT AFTER A RECORD EXCEEDS THE ERROR BOUND, RECORD
C*** THAT POINT AS WELL.
C
       DO 120 II=1,nssg
       IF(RI .EQ. RII+1.) BKSP(II) = CURNT(II)
  120  OUT(II) = BKSP(II)
C
       RI = RII
       PRMT(11) = PRMT(11) + 1.
C
       call ssout(out)
       RETURN
C
 1000  CONTINUE
C
C*** STOP INTEGRATION
C
```

```
      PRMT(12) = X
      PRMT(11) = PRMT(11) + 1.
C
      call ssout(curnt)
C
      PRMT(5) = 1.
C
      RETURN
      END
####
```

```
      subroutine ssout(out)
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      dimension out(9)
c
      common
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/com_fl/ cflag,clfl,cufl
     $/alp/ alpha,alpha1
     ./oomsin/ oodist,avtime
c
      character*3 gas_name
c
      data ip/0/
c
      logical cflag
c
c
      dist = out(1) + oodist
      yc = out(2)
      cc = out(3)
      rho = out(4)
      gamma = out(5)
      temp = out(6)
      b = out(7)
      sz = out(8)
      sy = out(9)
c
      if(.not.cflag) then
              call adiabat(-2,wc,wa,gas_lfl,ya,clfl,r,w,gamma,tt)
              call adiabat(-2,wc,wa,gas_ufl,ya,cufl,r,w,gamma,tt)
                endif
c
      arg = (gas_zsp/sz)**alpha1
      if(arg .ge. 80.) then
              if(cflag) then
      WRITE(8,1120) DIST,yc,Cc,rho,temp,B,SZ,SY
                      else
      WRITE(8,1125) DIST,yc,Cc,rho,gamma,temp,B,SZ,SY
                      endif
              goto 600
      endif
c
      ccz = cc/exp(arg)
      if(ccz .lt. clfl) then
              if(cflag) then
      WRITE(8,1120) DIST,yc,Cc,rho,temp,B,SZ,SY
                      else
```

```
          WRITE(8,1125) DIST,yc,Cc,rho,gamma,temp,B,SZ,SY
                      endif
                goto 600
                endif
          arg = -(dlog(clfl/cc) + (gas_zsp/sz)**alpha1)
          blfl = sqrt(arg)*sy + b
c
          if(ccz .lt. cufl) then
                if(cflag) then
          WRITE(8,1120) DIST,yc,Cc,rho,temp,B,SZ,SY,blfl
                      else
          WRITE(8,1125) DIST,yc,Cc,rho,gamma,temp,B,SZ,SY,blfl
                      endif
                goto 600
                endif
          arg = -(dlog(cufl/cc) + (gas_zsp/sz)**alpha1)
          bufl = sqrt(arg)*sy + b
                if(cflag) then
          WRITE(8,1120) DIST,yc,Cc,rho,temp,B,SZ,SY,blfl,bufl
                      else
          WRITE(8,1125) DIST,yc,Cc,rho,gamma,temp,B,SZ,SY,blfl,bufl
                      endif
c
  600     continue
          ip = ip + 1
          if(ip .eq. 3) then
                ip = 0
                write(8,1119)
                endif
C
C
 1119     FORMAT(1H )
 1120     FORMAT(1H ,11(1X,1PG9.3,1X))
 1125     FORMAT(1H ,3(1X,1PG9.3,1X),2x,0PF7.4,2x,1X,1PG10.3,1X,
          1      6(1X,1PG9.3,1X))
C
          return
          end
####
```

```
C..............................................................
C
C       PSEUDO-STEADY STATE SUPERVISOR
C
        SUBROUTINE SSSUP(H_masrte)
C

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADIS2.dec/nolist'
c
        COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/SSCON/ NREC(maxnob,2),TO(maxnob),XV(maxnob)
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $      PFRACV(igen), PENTH(igen), PRHO(igen)
     $/gen2/ den(5,igen)
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/ERROR/SYOER,ERRO,SZOER,WTAIO,WTQOO,WTSZO,ERRP,SMXP,
     $ WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WTDHG
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/STP/ STPO,STPP,ODLP,ODLLP,STPG,ODLG,ODLLG
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/nend/ poundn, pound
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     $/STOPIT/ TSTOP
     $/CNOBS/ NOBS
c
        REAL*8 K,ML,L
        LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
        logical pup,pdn
c
        character*4 pound
        character*3 gas_name
c
        EXTERNAL PSS,PSSOUT,SSG,SSGOUT,OB,OBOUT
C
        DIMENSION PRNT(22),Y(5),DERY(5),AUX(8,5)
C
        DATA RTOT/0./
        data wma/28.96/, wmw/18.0/
C
c
c*** Estimate the earliest and latest time an observer can be released
```

```
c***   over the source.
c
       R = AFGEN(RADG, 0.0D0, 'RADG')
       T01 = TOOB(R, 0.0D0)
c
c**  For low wind speed cases which form a blanket, earlier times
c**    than T01 may be possible.  Check each of the points in RADG.
c
       do i=2,maxl
       if ( radg(1,i).eq.poundn .and. radg(2,i).eq.poundn) goto 20
       TOF = TOob( radg(2,i), radg(1,i) )
       if ( TOF .lt. T01) T01 = TOF
       enddo
c
c**  Now, calculate the last possible time an observer can be released.
c
 20    continue
       XEND = AFGEN(RADG,TEND,'RADG')
       TOF = TOOB(-XEND,TEND)
C
c*** Now, divide the total possible time among the observers.
c
       DTOB = (TOF-T01)/FLOAT(NOBS)
       T01 = T01 + DTOB/2.D0
C
C*** perform the calculation for each observer
c
       write(12,1162)
c
       DO 120 I = 1,NOBS
C
C*** RESET AGAIN
C
       AGAIN = .FALSE.
C
       TO(I) = DTOB*dble(I-1) + T01
       pup = .true.
       pdn = .true.
C
C*** IF(XEND .GT. XIT(TEND,TO(I))) -- IS
C***  TRUE WHEN THE SOURCE WILL TERMINATE BEFORE THE OBSERVER
C***  CAN REACH THE DOWNWIND EDGE.
C
       IF(XEND .GT. XIT(TEND,TO(I))) then
              pdn = .false.
              TDOWN = TEND
              endif
c
C*** IF(XIT(0.0D0,TO(I)) .gt. -R) -- IS
C***  TRUE WHEN THE SOURCE WILL begin after THE OBSERVER
C***  has passed THE DOWNWIND EDGE @ t=0.0
```

```
C
      R = AFGEN(RADG,0.0D0,'RADG')
      IF(tO(i).le.0. .and. XIT(0.0D0,TO(I)).gt.-R) then
            pup = .false.
            TUP = 0.0D0
            endif
c
      if(pup) TUP = TUPF(TO(I))
      if(pdn) TDOWN = TDNF(TO(I))
C
      XDOWN = XIT(TDOWN,TO(I))
      XUP = XIT(TUP,TO(I))
      WRITE(lunlog,1160) TUP,XUP,TDOWN,XDOWN
C
C*** SET UP INTEGRATION PARAMETERS FOR EACH OBSERVER.
C
      do ijk=1,22
      prmt(ijk) =0.D0
      enddo
      do ijk=1,5
      y(ijk) = 0.D0
      dery(ijk)= 0.D0
      do ijkl=1,8
      aux(ijkl,ijk) = 0.D0
      enddo
      enddo
c
      PRMT(1) = TUP
      PRMT(2) = TDOWN
      PRMT(3) = STPO
      PRMT(4) = ERRO
      PRMT(5) = dMAX1(1.D0,(TDOWN-TUP)/50.D0)
      PRMT(6) = TO(I)
      prmt(7) = xup
      PRMT(13)= XDOWN - XUP
C
      Y(1) = sz0er    ! tos;3mar86
      Y(2) = sz0er    ! Mrate
      y(3) = sz0er * AFGEN(srcwc,tup,'IC')    ! Crate
      y(4) = sz0er * AFGEN(srcwa,tup,'IC') + 1.D-6    ! BDArate
      y(5) = sz0er * AFGEN(srcenth,tup,'IC')  ! Hrate
C
      DERY(1) = WTAIO
      DERY(2) = WTQOO
      DERY(3) = WTSZO
      DERY(4) = 1.D0
      DERY(5) = 1.D0
C
      NDIM = 4
      if(isofl.eq. 0 .and. ihtfl.ne. 0) ndim=5
C
```

```
C*** PERFORM INTEGRATION.
C
      WRITE(lunlog,1120) I
 1120 FORMAT(/,'  Entering Observer Integration Step for Observer # ',
     $  I3)
C
      CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,OB,OBOUT,AUX)
C
      IF(IHLF .GE. 10) CALL trap(8,IHLF)
c
      write(lunlog,1125)
 1125 format(' ',10x,'Observer Integration complete...')
c
c
c*** Establish initial conditions
c
      cclay = prmt(14)
      cc = cclay*dellay
      wclay = prmt(15)
      walay = prmt(16)
      enthlay=prmt(17)
      rholay = prmt(18)
C
      L = XDOWN - XUP
      B = Y(1)
      AREA = B*L
      QSTR0 = Y(3)/area
      Erate = 2.D0*qstr0*L*b
      cc = min(cc, rhoe)
      WCP = AFGEN2(PTIME,PWC,TDOWN,'SSS-WC')
      RHOP = AFGEN2(PTIME,PRHO,TDOWN,'SSS-RH')
      CCP = WCP*RHOP
      if(cc .gt. ccp) cc = ccp

      sz0 = (qstr0*L/cc * alpha1/u0/z0)**(1.D0/alpha1) * z0
C
      ratio1= u0*z0/ALPHA1/ Z0**ALPHA1 *Cc /B/qstr0/L
      ratio = ratio1* sz0**alpha1 * (B + sqrtpi/2.D0*sy0er)
      if(ratio.le. 1.D0) then
            sy0er = (1.D0/(RATIO1*sz0**alpha1) - b)*2.D0/sqrtpi
      else
            sz0 = (1.D0/((B+ sqrtpi/2.D0*sy0er)*ratio1))**(1.D0/alpha1)
      endif

c
c*** Establish the thermodynamic properties of mixtures of air and
c***  the gas mixture (WCLAY,WALAY,ENTHLAY) assuming adiabatic
c***  mixing.  This is accomplished with the call to SETDEN.
c***  Then, extrapolate the properties to the centerline,
c***  ground level concentration.
c
```

```
        humsrc = (1.D0-wclay-walay*(1.D0+humid))/wclay
        call setden(wclay,walay,enthlay)
        if(isofl.eq. 1) goto 200
c
c*** Scan through the array DEN for the last value, and establish a new
c***  final value based on the centerline, ground level concentration.
c
        do iii= 1,igen
        if(den(1,iii) .gt. 1.D0) then
               ii = iii+1
               rholay = den(3,iii-1)
               temlay = den(5,iii-1)
               if(ii .gt. igen .or. iii.le.3) call trap(2)
               cc = cclay*dellay        ! exact due to profile assumptions
        if(cc.gt. rhoe) then
          write(lunlog,1126) cc,rhoe
1126      format(/,' ',10('** **'),/,' cc: ',1pg13.5,' is greater',
     $         ' than rhoe: ',1pg13.5,/,' ',10('* ** *'),/)
          cc =rhoe
        endif
              rho = cc*(rholay-rhoa)/cclay + rhoa       ! assumes gamma=con
              wc = cc/rho
              w2 = den(2,iii-2)/den(3,iii-2)
              w1 = den(2,iii-1)/den(3,iii-1)
              wa = (1.D0-(1.D0+humsrc)*wc)/(1.D0+humid)
              wm = 1.D0/(wc/gas_mw + wa/wma + (1.D0-wa-wc)/wmw)
              yc = wm/gas_mw * wc
              Yc= dmax1(0.0D0, dmin1(1.0D0, yc) )
              den(1,iii) = Yc
              den(2,iii) = cc
              den(3,iii) = rho
c
c*** now, determine the enthalpy and temperature of such a mixture.
c***  Base both enthalpy and temperature on the fact that
c***  (yc/rho) is proportional to temperature (and therefore enthalpy).
c
              denom = den(1,iii-2)/den(3,iii-2)-den(1,iii-1)/den(3,iii-1)
      if(denom.ne. 0.D0) then
        slope = (yc/rho-den(1,iii-1)/den(3,iii-1))/denom
        den(4,iii) = slope*(den(4,iii-2)-den(4,iii-1))+den(4,iii-1)
        den(4,iii) = dmin1( dmax1(h_masrte,den(4,iii)), 0.D0)
        den(5,iii) = slope*(den(5,iii-2)-den(5,iii-1))+den(5,iii-1)
        den(5,iii) = dmin1( dmax1(gas_temp,den(5,iii)), tamb)
      else
        den(4,iii) = den(4,iii-1)
        den(5,iii) = den(5,iii-1)
      endif
             temp = den(5,iii)
             den(1,ii) = 2.D0         ! end-of-record
c.     if(cc.gt. rhoe) call trap(31)
             goto 200
```

```
                endif
          enddo
C
 200      CONTINUE

          IF(Cc .GT. RHOE) then
                WRITE(lunlog,1127) QSTRO,SZ0,Cc,rhoe
 1127           format(/,' ',10('****  '),/,' qstr0: ',1pg13.5,
     $          ' sz0: ',1pg13.5,/,
     $          /,' cc: ',1pg13.5,' is greater',
     $           ' than rhoe: ',1pg13.5,//,' ',10('****  '),/)
c               call trap(30)
                endif
C
C*** SHOW THE OPERATOR WHAT IS GOING ON
C
        WRITE(lunlog,1160) TUP,XUP,TDOWN,XDOWN
        WRITE(lunlog,1170) AREA,L,B
        WRITE(lunlog,1180) QSTRO,SZ0,sy0er
        write(lunlog,1185) wclay,walay,rholay,cclay,temlay
        write(lunlog,1186) wc,rho,cc,temp

        write(12,1161) i,t0(i), tup,tdown, xdown,L, B,Y(3), wc,temp,Sz0

 1160   FORMAT(/,' TUP: ',1pG13.5,' XUP: ',1pG13.5,' TDOWN: ',
     $ 1pG13.5,' XDOWN: ',1pG13.5)
 1161   format(' ',I3,1x,f8.1,1x,6(1x,f6.1,1x),1x,f5.3,1x,1x,
     $ f5.1,1x,1x,f5.2)
 1162   format(/,' ','obs',4x,'T0',4x,2x,'Tup',3x,1x,'Tdown',2x,
     $   1x,'Xdown',2x,1x,'Length',1x,1x,'HWidth',1x,1x,'E rate',1x,
     $   'Mass fr',1x,'Temp',2x,2x,'Sz0',//)
 1170   FORMAT(' Half AREA: ',1pG13.5,' LENGTH: ',1pG13.5,' B: ',1pG13.5)
 1180   FORMAT(' TAKEUP FLUX: ',1pG12.5,' SZ0: ',1pG12.5,
     $ ' sy0: ',1pg12.5)
 1185   format(' wclay: ',1pg12.5,' walay: ',1pg12.5,
     $          ' rholay: ',1pg12.5,' Cclay: ',1pg12.5,/,
     $          ' temlay: ',1pg13.5)
 1186   format(' wc: ',1pg12.5,
     $          ' rho: ',1pg12.5,' Cc: ',1pg12.5,' temp: ',1pg12.5)
C
C*** PREPARE FOR PSEUDO-STEADY STATE INTEGRATION.
C
        do ijk=1,22
        prmt(ijk) =0.D0
        enddo
        do ijk=1,5
        y(ijk) = 0.D0
        dery(ijk)= 0.D0
        do ijkl=1,8
        aux(ijkl,ijk) = 0.D0
        enddo
        enddo
c
        PRMT(1) = XDOWN
        PRMT(2) = 6.023D23
        PRMT(3) = STPP
        PRMT(4) = ERRP
        PRMT(5) = SMXP
        PRMT(6) = Erate
        PRMT(7) = Cc    ! -- OUTPUT
        PRMT(8) = B     ! -- OUTPUT
C
C*** PRMT(9) & PRMT(10) ARE CONSTANTS FOR D(SY) & D(SZ)
C
        PRMT(9) = Ce*sqrt(G*Z0/ALPHA1*GAMMAF)*GAMMAF/U0
        PRMT(10)= Z0**ALPHA*K*USTAR*ALPHA1*ALPHA1/U0
        PRMT(11)= NREC(I,1)
C       PRMT(12)= DIST AT COMPLETION -- OUTPUT
        PRMT(13)= TO(I)
c       prmt(14)= yc    ! output
c       prmt(15)= rho   ! output
c       prmt(16)= temp  ! output; not recorded if isofl=1
c       prmt(17)= gamma ! output; not recorded if isofl=1 .or. ihtfl=0
        prmt(18)= u0*z0/alpha1
        prmt(19)= rhoa*k*ustar*alpha1
        prmt(20)= rholay
        prmt(21)= sz0
        prmt(22)= sz0
C
        Y(1) = rholay*prmt(18)*(SZ0/z0)**alpha1           ! rho*ueff*heff
        Y(2) = SYOER*SYOER
        Y(3) = B + sqrtpi/2.D0*sy0er
        y(4) = 0.                      ! added heat
C
        DERY(1) = WTSZP
        DERY(2) = WTSYP
        DERY(3) = WTBEP
        dery(4) = WTDH
C
        NDIM = 4
C
        WRITE(lunlog,1130)
 1130   FORMAT('  Entering Integration Step -- B > 0. ')
C
C*** PERFORM INTEGRATION
C
        CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,PSS,PSSOUT,AUX)
C
        IF(IHLF .GE. 10) CALL trap(9,IHLF)
C
        NREC(I,1) = INT(PRMT(11))
        WRITE(lunlog,1100) NREC(I,1),TO(I)
 1100   FORMAT(3X,'NUMBER OF RECORDS IN PSS = 'I10,' FOR TO='1pg13.5)
C
      IF(AGAIN) GO TO 119
C
C*** GAUSIAN COMPLETION OF THE INTEGRATION
C
C*** PSSOUT FORCES THE ABOVE INTEGRATION TO FINISH WHEN B<0 FOR THE
C***  FIRST TIME. THE STEP BEFORE THIS OCCURS IS RECORDED ON UNIT 7.
C***  THE STEP WHEN B GOES NEGATIVE IS CURRENTLY IN Y.
C
C*** THE CALCULATION METHOD CHANGES THE CURRENT VALUE OF SY TO A VALUE
C***  CALCULATED AS IF BEFF=sqrtpi*SY/2. RETAINING THE LAST VALUE OF Cc IN THE
C***  MATERIAL BALANCE.
C
      heat = y(4)
      rholay = prmt(20)
      Cc = PRMT(7)
      rhouh = Y(1)
      sz = ( rhouh/rholay/prmt(18) )**(1.D0/alpha1) * z0
      SYT = Erate*ALPHA1*(Z0/SZ)**ALPHA/U0/SZ/Cc/SQRTPI
C
      XT = PRMT(12)
      XV(I) = (SYT/RT2/DELTA)**(1.D0/BETA) - XT
C
C*** SET UP INTEGRATION FOR THE GAUSSIAN DISPERSION PHASE.
C
      do ijk=1,22
      prmt(ijk) =0.D0
      enddo
      do ijk=1,5
      y(ijk) = 0.D0
      dery(ijk)= 0.D0
      do ijkl=1,8
      aux(ijkl,ijk) = 0.D0
      enddo
      enddo
c
      PRMT(1) = XT
      PRMT(2) = 6.023D23
      PRMT(3) = STPG
      PRMT(4) = ERRG
      PRMT(5) = SMXG
      PRMT(6) = Erate
      PRMT(7) = Cc    ! -- OUTPUT
      PRMT(8) = XV(I)
      PRMT(9) = TO(I)
C     PRMT(10)= "BLANK"
      PRMT(11)= NREC(I,2)
C     PRMT(12)= DIST AT COMPLETION -- OUTPUT
c     prmt(13)= "blank"
c     prmt(14)= yc          ! output
```

```
c       prmt(15)= rho            ! output
c       prmt(16)= temp           ! output
c       prmt(17)= gamma          ! output
        prmt(18)= u0*z0/alpha1
        prmt(19)= rhoa*k*ustar*alpha1
        prmt(20)= rholay
        prmt(21)= sz
        prmt(22)= sz
c
        Y(1) = rhouh
        Y(2) = heat
c
        DERY(1) = WTRUH
        dery(2) = WTDHG
c
        NDIM = 2
c
        WRITE(lunlog,1140)
 1140   FORMAT(' Entering Gaussian Stage of Integration ')
c
C*** PERFORM INTEGRATION
c
        CALL RKGST(PRMT,Y,DERY,NDIM,IHLF,SSG,SSGOUT,AUX)
c
        IF(IHLF .GE. 10) CALL trap(10,IHLF)
c
        NREC(I,2) = INT(PRMT(11))
        RTOT = RTOT + FLOAT(NREC(I,1) + NREC(I,2))
        WRITE(lunlog,1110) RTOT,I
 1110   FORMAT(5X,'TOTAL NUMBER OF RECORDS = ',1pG13.4,' THROUGH',
     $' OBS # ',I3)
c
        IF(RTOT .GT. 120000.) CALL trap(11)
c
  119   CONTINUE
        write(lunlog, 1150) tstop, Prmt(12)
 1150   format(/,'  Last time Observer was active: ',1pg13.5,' s at ',
     $   1pg13.5,' m')
  120   CONTINUE
c
        RETURN
        END
####
```

```
C...............................................................
C
      SUBROUTINE STRT2(OPNRUP,H_masrte)
C

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS2.dec'
c
      COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/TITL/ TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/ERROR/SYOER,ERRO,SZOER,WTAIO,WTQOO,WTSZO,ERRP,SMXP,
     $ WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WTDHG
     $/PARM/ UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     $/COM_SURF/ HTCUT
     ./oomsin/ oodist,avtime
c
      character*80 TITLE(4)
c
      character*4 pound
      character*24 TINP,TSRC,TOBS,TSRT
      character*3 gas_name
c
      REAL*8 K,ML
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
c
      character*40 OPNRUP
C
      OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
      DO 90 I = 1,4
   90 READ(9,1000) TITLE(I)
 1000 FORMAT(A80)
C
```

```
        READ(9,*) NP
        DO 100 I=1,NP
  100   READ(9,*) PTIME(I),ET(I),R1T(I), PWC(I), PTEMP(I),
     $       PFRACV(I), PENTH(I), PRHO(I)
        PTIME(NP + 1)  = POUNDN
c
        READ(9,*) NP
        DO 220 I=1,NP
  220   READ(9,*) DEN(1,I),DEN(2,I),den(3,I),den(4,i),den(5,i)
        den(1,np+1) = 2.
c
        READ(9,*) NP
        DO 300 I=1,NP
        READ(9,*) radg(1,I),radg(2,I),qstr(2,I),srcden(2,I),srcwc(2,i),
     1       srcwa(2,i),srcenth(2,i)
        qstr(1,I) = radg(1,I)
        srcden(1,I) = radg(1,I)
        srcwc(1,i) = radg(1,i)
        srcwa(1,i) = radg(1,i)
        srcenth(1,i) = radg(1,i)
  300   continue
        I = NP + 1
        radg(1,I)  = POUNDN
        radg(2,I)  = POUNDN
        qstr(1,I) = POUNDN
        qstr(2,I) = POUNDN
        srcden(1,I) = POUNDN
        srcden(2,I) = POUNDN
        srcwc(1,I) = POUNDN
        srcwc(2,I) = POUNDN
        srcwa(1,I) = POUNDN
        srcwa(2,I) = POUNDN
        srcenth(1,I) = POUNDN
        srcenth(2,I) = POUNDN
c
        READ(9,1010) TINP,TSRC
        READ(9,1010) tobs,TSRT
c
        read(9,*) oodist,avtime
c
        READ(9,*) U0,Z0,ZR,ML,USTAR
        read(9,*) K,G,RHOE,RHOA,DELTA
        read(9,*) BETA,GAMMAF,CcLOW
c
        READ(9,*) RM,SZM,EMAX,RMAX,TSC1
        read(9,*) ALEPH,TEND
c
        READ(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
c
        READ(9,*) ALPHA
        alpha1 = alpha + 1.
```

```
c
        read(9,1020) gas_name
        read(9,*) gas_mw,gas_temp,gas_rhoe
        read(9,*) gas_cpk,gas_cpp
        read(9,*) gas_ufl,gas_lfl,gas_zsp
c
        read(9,*) istab
        read(9,*) tamb,pamb,humid
        humsrc = 0.D0
        read(9,*) isofl,tsurf
        read(9,*) ihtfl,htco
        read(9,*) iwtfl,wtco
c
        read(9,*) sigx_coeff,sigx_pow,sigx_min_dist
c
        read(9,*) iphifl,dellay
c
        H_masrte = 0.
        if(isofl.eq. 0) read(9,*) H_masrte
C
        READ(9,*) HTCUT, ce, delrhomin
c
 1010   format(2(a24,1x))
 1020   format(a3)
C
        CLOSE(UNIT=9)
C
        RETURN
        END
####
```

```
C..............................................................
C
      SUBROUTINE STRT2(OPNRUP,H_masrte,CCP)
C
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      INCLUDE 'sys$degadis:DEGADIS2.DEC/LIST'
C
C..............................................................
C
C     BLOCK COMMON
C
      COMMON
     $/TITL/ TITLE
     $/GEN2/ DEN(5,IGEN)
     $/ITI/ T1,TINP,TSRC,TOBS
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_ss/ ESS,SLEN,SWID,OUTCc,OUTSZ,OUTB,OUTL,swcl,swal,senl,srhl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     $/COM_SURF/ HTCUT
     ./comsin/ oodist,avtime
C
      character*80 TITLE(4)
      character*24 TSRC,TINP,TOBS
      character*40 OPNRUP
      character*3 gas_name
      character*4 pound
C
      REAL*8 K,ML
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
      OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
      DO 90 I = 1,4
   90 READ(9,1000) TITLE(I)
 1000 FORMAT(A80)
C
      read(9,*) np
      do 100 i=1,np
      read(9,*) dummy1,dummy2,dummy3,PWC,PTEMP,PFRACV,PENTH,PRHO
  100 IF(I .EQ. 1) CCP = PWC*PRHO
c
      READ(9,*) NP
      DO 120 I=1,NP
  120 READ(9,*) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
      I = NP + 1
      DEN(1,I) = 2.
c
      read(9,*) np
      do 140 i=1,np
  140 read(9,*) dummy1,dummy2,dummy3,dummy4,dum5,dum6,dum7
C
      read(9,1100) tinp,tsc
      read(9,1100) tobs,tsrt
 1100 format(a24,1x,a24)
c
      read(9,*) oodist,avtime
c
      READ(9,*) U0,Z0,ZR,ML,USTAR
      Read(9,*) K,G,RHOE,RHOA,DELTA
      read(9,*) BETA,GAMMAF,CcLOW
c
      read(9,*) dummy1
      read(9,*) dummy1
c
      READ(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
c
      READ(9,*) ALPHA
      alpha1 = alpha + 1.
c
      read(9,1200) gas_name
      read(9,*) gas_mw,gas_temp,gas_rhoe
      read(9,*) gas_cpk,gas_cpp
      read(9,*) gas_ufl,gas_lfl,gas_zsp
c
      read(9,*) istab
      read(9,*) tamb,pamb,humid
      humsrc = 0.D0
      read(9,*) isofl,tsurf
      read(9,*) ihtfl,htco
      read(9,*) iwtfl,wtco
      read(9,*) dummy1

      READ(9,*) ESS,SLEN,SWID
      read(9,*) OUTCc,OUTSZ,OUTB,OUTL
      read(9,*) swcl,swal,senl,srhl
C
      read(9,*) iphifl,dellay
      h_masrte = 0.
      if(isofl.eq. 0) read(9,*) H_masrte
C
      READ(9,*) HTCUT, ce, delrhomin
      CLOSE(UNIT=9)
      RETURN
```

```
1200   format(a3)
       END
####
```

```
C...............................................................
C
        SUBROUTINE STRT3(OPNRUP)
C

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADIS3.dec/list'
C
C       BLOCK COMMON
C
        COMMON
     $/SSCON/ NREC(maxnob,2),TO(maxnob),XV(maxnob)
     $/GEN2/ DEN(5,igen)
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/CNOBS/ NOBS
     ./oomsin/ oodist,avtime
C
        character*3 gas_name
        character*40 OPNRUP
        character*24 TINP,TSRC,TOBS,TSRT
C
        REAL*8 K,ML
        LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
        OPEN(UNIT=9,NAME=OPNRUP,TYPE='OLD')
C
        READ(9,*) NOBS
        DO 125 I=1,NOBS
  125   READ(9,*) NREC(I,1),NREC(I,2),TO(I),XV(I)
c
        READ(9,*) Npts
        DO 140 I=1,Npts
  140   READ(9,*) den(1,i),den(2,i),den(3,i),den(4,i),den(5,i)
        den(1,npts+1) = 2.
c
        read(9,*) oodist,avtime
c
        READ(9,*) U0,Z0,ZR,ML,USTAR
        read(9,*) K,G,RHOE,RHOA,DELTA
        read(9,*) BETA,GAMMAF,CcLOW
```

```
c
        READ(9,1010) TINP,TSRC
        read(9,1010) TOBS,TSRT
 1010   format(2(a24,1x))
c
        READ(9,*) RM,SZM,EMAX,RMAX,TSC1
        read(9,*) ALEPH,TEND
c
        read(9,1020) gas_name
        read(9,*) gas_mw,gas_temp,gas_rhoe
        read(9,*) gas_cpk,gas_cpp
        read(9,*) gas_ufl,gas_lfl,gas_zsp
        read(9,*) istab
        read(9,*) tamb,pamb,humid
        humsrc = 0.D0
        read(9,*) isofl,tsurf
        read(9,*) ihtfl,htco
        read(9,*) iwtfl,wtco
        read(9,*) sigx_coeff,sigx_pow,sigx_min_dist
 1020   format(a3)
c
        READ(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
        READ(9,*) ALPHA
        alpha1 = alpha + 1.
C
        CLOSE(UNIT=9)
C
        RETURN
        END
####
```

```
C      Surface effects
C
       SUBROUTINE Surface(temp,height,rho,mole,cp,yw,watrte,qrte)
C

       Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

       include 'sys$degadis:DEGADIS1.dec'
c
C
       COMMON
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/COM_SURF/ HTCUT
C
c
       REAL*8 ML,K
       REAL*8 L,masrte,mole
C
       vapor_p(txxx) = 6.0298e-3* exp(5407. *(1./273.15- 1./txxx))
c
c
c
       watrte = 0.
       qrte = 0.
       if(isofl.eq.1 .or. ihtfl.eq.0) return
       if(height.le. htcut) return
       delta_t = tsurf - temp

       if(delta_t .lt. 0.) return
       hhh = dmax1( height/2.D0, zr)
       top_vel = u0 * (hhh/z0)**alpha
       prod_nat = ((rho/mole)**2 * abs(delta_t)) ** 0.333333
       if(ihtfl .eq. 1) then              ! local correlation
              hn = 18. * prod_nat
              hf = 0.
              hf = 1.22 * rho*cp * ustar**2/top_vel
              ho = dmax1(hn,hf)
       else if(ihtfl .eq. 2) then             ! LLNL correlation
              ho = htco* rho* cp
       else if(ihtfl .eq. 3) then             ! Colenbrander's method
              av_temp= (tsurf + temp)/2.
              hn = 89.*(delta_t/av_temp**2)**.33333
              u10 = u0*(10./z0)**alpha
              hf = 1.22 * rho*cp * ustar**2/u10
              ho = dmax1(hn,hf)
       else
              ho = htco       ! ihtfl=-1
```

```
          endif
          qrte = ho * delta_t
          if(qrte .lt. 0.) qrte = 0.
  c                                  ! since correlations are not valid for qrte<0.
          watrte = 0.
          if(iwtfl .eq. 0) return
          fo = wtco
                  if(iwtfl .gt. 0) then
                  fn = 9.9e-3 * prod_nat
                  ff = 20.7 * ho /cp /mole
                  fo = dmax1(fn,ff)
                  endif
          watrte = min( vapor_p(temp), yw*pamb )
          watrte = fo * (vapor_p(tsurf) - watrte)/pamb
          if(watrte .le. 0.) watrte = 0.
  c
          return
          end
####
```

```
C
C       FUNCTION TO RETURN SZ0 CALCULATED over the source without
c        a blanket present underneath
c
c       NOTE: Uses the integration package RKGST and cannot be used
c           with any other routine without a local copy of RKGST.
C
        subroutine SZF(Q,L,WCP,sz,cclay,wclay,rholay)
c
        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        external szlocal,szloco
C
        REAL*8 L
C
        include 'sys$degadis:DEGADIS1.dec'
C
        COMMON
     $/szfc/ szstp0,szerr,szstpmx,szsz0
C
        dimension Y(1),D(1),PRMT(17),aux(8,1)
c
        prmt(1) = 0.
        prmt(2) = L
        prmt(3) = szstp0
        prmt(4) = szerr
        prmt(5) = szstpmx
        prmt(6) = Q
        PRMT(7) = WCP
c
        Y(1) = 0.D0     ! rho*dellay*u0*z0/(1.+alpha)*(sz/z0)**(1.+alpha)
        D(1) = 1.D0
c
        ndim = 1
c
        call rkgst(prmt,y,d,ndim,ihlf,szlocal,szloco,aux)
c
        if(ihlf.ge. 10) call trap(3,ihlf)
c
        cclay = prmt(13)
        wclay = prmt(14)
        rholay = prmt(15)
        cc = prmt(16)
        sz = prmt(17)
C
        RETURN
        END
c
c
        subroutine szlocal(x,y,d,prmt)
c
```

```
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      dimension y(1),d(1),prmt(1)
c
      common
     $/parm/ u0,z0,zr,ml,ustar,k,g,rhoe,rhoa,delta,beta,gammaf,cclow
     $/alp/ alpha,alpha1
     $/phicom/ iphifl,dellay
c
      real*8 ml,k
      integer rhouhlay/1/
c
      Q = prmt(6)
      WCP = PRMT(7)
c
c... for start up...
c
      if(Y(rhouhlay) .le. 0.D0) then
            wclay  = 0.D0
            rholay = rhoa
            cclay  = 0.D0
            cc     = 0.D0
            rho    = rhoa
      else
c
c... from the contaminant material balance over the source, determine WCLAY
c
            wclay = Q*x/Y(rhouhlay)
c
            call adiabat(1,wclay,walay,yc,ya,cclay,rholay,wm,enth,temp)
            cc = cclay*dellay
            call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enth,temp) ! centerline
c
      endif
c
      uheff = Y(rhouhlay)/rholay/dellay
      sz = ( uheff/u0/z0*(alpha1) )**(1./alpha1) * z0
      heff = gammaf*sz/alpha1
      ristar = rif(rho,heff)
      phi = phif(ristar,0.D0)
      wel = dellay * k*ustar*alpha1/phi
      D(rhouhlay) = wel*rhoa + Q/WCP
c
      prmt(8) = cclay
      prmt(9) = wclay
      prmt(10)= rholay
      prmt(11)= cc
      prmt(12) = sz
      return
      end
c
```

```
c
      subroutine szloco(x,y, dery,ihlf,ndim, prmt)
c
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      dimension y(1), dery(1), prmt(1)
c
      prmt(13) = prmt(8)
      prmt(14) = prmt(9)
      prmt(15) = prmt(10)
      prmt(16) = prmt(11)
      prmt(17) = prmt(12)
      return
      end
####
```

```
      subroutine tprop(ifl,wc,wa,enthalpy,yc,ya,wm,temp,rho,cp)
c
c     subroutine to return:
c             mole fractions (y's)
c             molecular weight (wm)
c             temperature (temp[=]K)
c             density (rho[=]kg/m**3)
c             heat capacity (cp[=]J/kg/K)
c
c     for a mixture from:
c             mass fractions (w's)
c             temperature (K)         for ifl.lt.0
c
c     for a mixture from:
c             mass fractions (w's)
c             enthalpy (J/kg)         for ifl.ge.0
c
c             adiabatic mixing of:    emitted gas @ gas_temp
c                                     entrained ambient humid air @ tamb
c                                     entrained water from surface @ tsurf
c                             for ifl.eq.0 calculate and return
c
c             adiabatic lookup CALL ADIABAT
c                             for isofl.eq.1 .or. ihtfl.eq.0.and.ifl.eq.1
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADIS1.dec'
c
      parameter (acrit=0.02D0)
c
      common
     $/GEN2/ DEN(5,igen)
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     ./ctprop/ cwc,cwa,centh
c
      character*3 gas_name

      external enthal0
c
c*** data for air/water sys
c
      data wma/28.96D0/       ! molecular weight of air
      data wmw/18.02D0/       ! molecular weight of water
      data rho_water/1000.D0/ ! liquid water density [=] kg/m**3
      data cpa/1.0063D3/      ! heat capacity of air [=]J/kg/K
      data cpw/1865.D0/       ! heat capacity of water vapor[=]J/kg/K
      data dhvap/2.5023D6/    !latent heat of vap [=]J/kg water
      data dhfus/0.33D6/      !latent heat of fus [=]J/kg water

      data rgas/0.08205D0/    ! gas constant for P[=]atm, T[=]K
c
c
      vapor_p(txxx) = 6.0298D-3* exp(5407.D0 *(1.D0/273.15D0- 1.D0/txxx))
c
c
      ww = 1.D0-wc-wa
      wm = 1.D0/(wc/gas_mw + wa/wma + ww/wmw)
      yc = wm/gas_mw *wc
      ya = wm/wma *wa
      yw = 1.D0 - yc - ya
c
c
      if(isofl.eq. 1) then
              call adiabat(1,wc,wa,yc,ya,cc,rho,wm,enthalpy,temp)
              return            ! interp density from wc
              endif
c
c
      if(ifl.eq. 0)
     $        enthalpy = wc*cpc(gas_temp)*(gas_temp - tamb)
     $        + (ww - wa*humid)*cpw*(tsurf - tamb)
c    $        + wa*humid*cpw*(tamb - tamb)
c    $        + wa*cpa*(tamb - tamb)            ! TR=tamb
c
c
c
      if(ifl.eq. 1 .and. ihtfl.eq.0) then
              call adiabat(1,wc,wa,yc,ya,cc,rho,wm,enthalpy,temp)
              return            ! interp density from wc
              endif
c
c
      if(ifl .eq. -1) goto 400
c
c
      tmin = dmin1(gas_temp, tsurf, tamb)
      tmax = dmax1(gas_temp, tsurf, tamb)

      elow = enthal(wc,wa,tmin)
      if(enthalpy.lt. elow) then
          temp = tmin
          enthalpy = elow
          goto 400
      endif

      elow = enthal(wc,wa,tmax)
      if(enthalpy.gt. elow) then
```

```
                temp = tmax
                enthalpy = elow
                goto 400
             endif
c
             cwc = wc
             cwa = wa
             centh = enthalpy

             call zbrent(temp, enthal0, tmin, tmax, acrit, ierr)
             if(ierr .ne. 0) call trap(24)
c
c
c
 400         continue                    ! density calculation
             vp = vapor_p(temp)
             ywsat = dmax1( vp/pamb, 0.D0)
             wwsat = wmw/wm * ywsat * (1.D0 - (yw-ywsat))
             conden = dmax1( 0.D0, ww-wwsat)

             rho = 1.D0/(       wa / (pamb*wma/rgas/temp)
     *                        + (ww-conden) / (pamb*wmw/rgas/temp)
     *                        + conden / rho_water
     *                        + wc * temp/gas_temp/gas_rhoe )
c
             tmin = temp + 10.
             if(tmin .gt. tmax0) tmin = temp - 10.
             if(tmin .lt. tmin0) tmin = temp + .1
c
             tmax = enthal(wc,wa,tmin)
             cp = (enthalpy - tmax)/(temp - tmin)
             if(cp .lt. cpa) cp = cpa                       ! nominal value of air
c
             return
             end
c
c
             function cpc(temp)
c

             Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

             common
         $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
         $ gas_ufl,gas_lfl,gas_zsp,gas_name
c
             data con/3.33D4/
c
             character*3 gas_name
c
             cpc = con + gas_cpk*gas_cpp * gas_temp**(gas_cpp-1.D0)
```

```
          if(temp .ne. gas_temp) then
                cpc = con + gas_cpk*
     .    (temp**gas_cpp - gas_temp**gas_cpp)/(temp- gas_temp)
                endif
          cpc = cpc/gas_mw
          return
          end
c
c
          function enthal(wc,wa,temp)     ! used by TPROP
c


          Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

          parameter (delta=10.D0)
c
          common
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/prop/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
c
          character*3 gas_name
c
          data cpa/1006.3D0/        ! heat capacity of air [=]J/kg/K
          data cpw/1865.D0/         ! heat capacity of water vapor[=]J/kg/K
          data dhvap/2.5023D6/      !latent heat of vap [=]J/kg water
          data dhfus/0.33D6/        !latent heat of fus [=]J/kg water
          data wma/28.96D0/         ! molecular weight of air
          data wmw/18.02D0/         ! molecular weight of water
c
c
          ww = 1.D0-wa-wc
          wm = 1.D0/(wc/gas_mw + wa/wma + ww/wmw)
          yw = ww * wm/wmw

          vp = 6.0298D-3* exp(5407.D0 *(1.D0/273.15D0- 1.D0/temp))
c
          dh = dhvap
          frac = 0.
          if(temp .lt. 273.15D0) frac = dmin1( (273.15D0-temp)/delta,1.D0)
          dh = dhvap + dhfus*frac
          ywsat = vp/pamb
          wwsat = wmw/wm * ywsat * (1.D0 - (yw-ywsat))
          conden = dmax1( 0.D0, ww-wwsat)
c
 1000     enthal = wc*cpc(temp)*(temp - tamb)
     $          - conden*dh
     $          + ww*cpw*(temp - tamb)
     $          + wa*cpa*(temp - tamb)         ! TR=tamb
```

```
c
        return
        end
c
c
        function enthal0( temp )
c
        implicit real*8 (a-h, o-z), integer*4(i-n)
c
        include 'sys$degadis:degadisin.dec'

        common /ctprop/ cwc,cwa,centh

c

        enthal0 = centh - enthal(cwc,cwa,temp)
        return
        end
c
c

        subroutine adiabat(ifl,wc,wa,yc,ya,cc,rho,wm,enthalpy,temp)
c
c       subroutine to return:
c               mass fractions (w's)
c               mole fractions (y's)
c               concentration (cc[=]kg/m**3)
c               density (rho[=]kg/m**3)
c               molecular weight (wm)
c               enthalpy ([=]J/kg)
c               temperature (temp[=]K)
c
c       for a mixture from DEN lookup of adiabatic mixing calculation
c       den(1,i)        mole fraction (yc)
c       den(2,i)        concentration (cc [=] kg c/m**3)
c       den(3,i)        mixture density (rho [=] kg mix/m**3)
c       den(4,i)        mixture enthalpy (enthalpy [=] J/kg)
c       den(5,i)        mixture temperature (temp [=] K)
c
c       ifl indicates given information:
c       -2)mole fraction (Yc) and assumption of constant gamma in enthalpy
c       -1)concentration (cc) and assumption of constant gamma in enthalpy
c       0) concentration (cc)
c       1) mass fraction c (wc)
c       2) mole fraction (Yc)
c

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADISIN.dec'
c
        common
     $/GEN2/ DEN(5,igen)
     $/parm/ u0,z0,zr,ml,ustar,k,g,rhoe,rhoa,delta,beta,gammaf,cclow
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
c
        character*3 gas_name
        real*8 ml,k
c
c*** data for air/water sys
c
        data wma/28.96D0/               ! molecular weight of air
        data wmw/18.02D0/               ! molecular weight of water
c
c
        if(ifl.ne. 0) goto 1000
        ccl = cc
        if(cc .lt. 0.) ccl=0.
                i = 2
 30             if(den(1,i) .gt. 1.) then
                        i=i-1
                        if(cc.gt. den(2,i)) ccl=den(2,i)
                        goto 50
                        endif
                if(cc.le. den(2,i)) goto 50     ! lookup in concentration
                i=i+1
                goto 30
 50     slope = (den(3,i)-den(3,i-1)) / (den(2,i)-den(2,i-1)) ! interp in conc
                rho = (ccl - den(2,i-1))*slope + den(3,i-1)
        wcl = ccl / rho
        wc = wcl
        wa = (1.D0-(1.D0+humsrc)*wc)/(1.D0+humid)
        ww = 1.D0 - wa - wc
        wm = 1.D0/(wc/gas_mw + wa/wma + ww/wmw)
        yc = wm/gas_mw * wc
        ya = wm/wma * wa
        goto 8000
c
c
 1000   if(ifl.ne. -1) goto 1500
        ccl = cc
        if(ccl.lt. 0.) ccl=0.
        gamma = enthalpy
        wc = ccl/(rhoe+ccl*gamma)
        wa = (1.D0-(1.D0+humsrc)*wc)/(1.D0+humid)
        ww = 1.D0-wa-wc
        wm = 1.D0/(wc/gas_mw + wa/wma + ww/wmw)
        yc = wm/gas_mw * wc
        ya = wm/wma * wa
        rho = ccl/wc
        return
```

```
c
c
 1500   if(ifl.ne. -2) goto 1700
        ycl = yc
        if(ycl.lt. 0.) ycl=0.D0
        gamma = enthalpy
        ya = (1.D0-(1.D0+gas_mw*humsrc/wmw)*ycl)/(1.D0+humid*wma/wmw)
        yw = 1.D0-ya-ycl
        wm = ycl*gas_mw + ya*wma + yw*wmw
        wc = gas_mw/wm * ycl
        wa = wma/wm * ya
        cc = wc*rhoa/(1.D0 - gamma*wc)
        rho = cc/wc
        return
c
c
 1700   if(ifl.ne. 2) goto 2000
        ycl = yc
        if(yc .lt. 0.) then
            ycl = 0.
            wa = 1.D0/(1.D0+humid)
            ww = 1.D0-wa
            wm = 1.D0/(wma/wa + wmw/ww)
            ya = wm/wma * wa
            endif
        if(yc .gt. 1.) then
            ycl = 1.D0
            ya = 0.D0
            endif
        i = 2
 1730   if(den(1,i) .gt. 1.) then
                i = i-1
                goto 1750                   ! extrapolate
                endif
        if(ycl.le. den(1,i)) goto 1750  ! lookup in mole frac
        i=i+1
        goto 1730
 1750   continue
        wm = ycl*gas_mw + (1.D0-ycl)*wma*wmw*(1.D0+humid) / (wmw + wma*humid)
        wc = ycl*gas_mw / wm
        wa = (1.D0-wc)/(1.D0 + humid)
        ww = 1.D0 - wc - wa

        slope = (den(3,i)-den(3,i-1)) / (den(2,i)-den(2,i-1))   ! interp in cc

        cc = wc*(den(3,i-1) - slope*den(2,i-1))/(1.D0 - wc*slope)
        rho = cc/wc

                w1 = den(2,i-1)/den(3,i-1)
                w2 = den(2,i)/den(3,i)
        slope = (den(4,i)-den(4,i-1)) / (w2 - w1)        ! interp in w
```

```
            enthalpy = (wc - w1) *slope + den(4,i-1)
        slope = (den(5,i)-den(5,i-1)) / (w2 - w1)       ! interp in w
            temp = (wc - w1) *slope + den(5,i-1)
c
        return
c
c
 2000   if(ifl.ne. 1) goto 9000
        wcl = wc
          if(wc .lt. 0.) then
            wcl = 0.D0
            wa = 1.D0/(1.D0+humid)
            endif
          if(wc .gt. 1.) then
            wcl = 1.D0
            wa = 0.D0
            endif
        ww = 1.D0-wa-wcl
        wm = 1.D0/(wcl/gas_mw + wa/wma + ww/wmw)
        yc = wm/gas_mw *wcl
        ya = wm/wma *wa
        i = 2
 2030   if(den(1,i) .gt. 1.) then
            i = i-1
            goto 2050                  ! extrapolate
            endif
        if(yc.le. den(1,i)) goto 2050  ! lookup in mole frac
        i=i+1
        goto 2030
 2050   slope = (den(3,i)-den(3,i-1)) / (den(1,i)-den(1,i-1))
        rho = (yc-den(1,i-1))*slope + den(3,i-1)
        slope = (den(2,i)-den(2,i-1)) / (den(1,i)-den(1,i-1))
        cc = (yc-den(1,i-1))*slope + den(2,i-1)
        i = 2
 2060   if(den(1,i) .gt. 1.) then
            i = i-1
            goto 8000                  ! extrapolate
            endif
        cwc = den(2,i)/den(3,i)
        if(wcl.le. cwc) goto 8000      ! lookup in mass frac
        i=i+1
        goto 2060
c
c
 8000       w1 = den(2,i-1)/den(3,i-1)
            w2 = den(2,i)/den(3,i)
        slope = (den(4,i)-den(4,i-1)) / (w2 - w1)       ! interp in w
            enthalpy = (wcl - w1) *slope + den(4,i-1)
        slope = (den(5,i)-den(5,i-1)) / (w2 - w1)       ! interp in w
            temp = (wcl - w1) *slope + den(5,i-1)
c
```

```
      return
c
 9000 call trap(26)
      end
c
c
      subroutine setenthal(h_masrte,h_airrte,h_watrte)
c
c     subroutine to load /com_ENTHAL/ through passed arguments if needed
c
c

      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

      include 'sys$degadis:DEGADISIN.dec'
c
      common
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
c
      character*3 gas_name
c
c*** data for air/water sys
c
      data cpa/1.0063D3/       ! heat capacity of air [=]J/kg/K
      data cpw/1865.D0/        ! heat capacity of water vapor[=]J/kg/K
c
      h_masrte = 0.D0
      h_airrte = 0.D0
      h_watrte = 0.D0
c
      if(isofl.eq. 1) return
c
      h_masrte = cpc(gas_temp)*(gas_temp - tamb)      ! TR=tamb
c
c h_airrte = cpa*(tamb - tamb) = 0.
c
      if(iwatfl .eq. 0) return
      h_watrte = cpw*(tsurf - tamb)
c
      return
      end
c
c
c
      subroutine setden(wc,wa,enthalpy)
c
c     subroutine to load /GEN2/ as needed
c
```

```
c       adiabatic mixing of:    WC
c                               WA
c                               WW @ specified enthalpy
c
c               with ambient humid air @ tamb
c
c       den(1,i)        mole fraction (yc)
c       den(2,i)        concentration (cc [=] kg c/m**3)
c       den(3,i)        mixture density (rho [=] kg mix/m**3)
c       den(4,i)        mixture enthalpy (enthalpy [=] J/kg)
c       den(5,i)        mixture temperature (temp [=] K)
c
c

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADISIN.dec'
c
        parameter (tcrit=0.002D0, zero=1.D-20)
        parameter (iils=200, ils=iils-1, iback=25)
c
        common
     $/GEN2/ DEN(5,igen)
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
c
        character*3 gas_name
c
        dimension curnt(5),backsp(5,iback)
c
c*** data for air/water sys
c
        data wma/28.96D0/       ! molecular weight of air
        data wmw/18.02D0/       ! molecular weight of water
        data cpa/1.0063D3/      ! heat capacity of air [=]J/kg/K
        data cpw/1865.D0/       ! heat capacity of water vapor[=]J/kg/K
c
c
c
        if(isofl.eq. 1) return
c
c
        k = 1
        den(1,k) = 0.0D0                ! yc
        den(2,k) = 0.0D0                ! cc
        den(3,k) = pamb*(1.D0+humid)/(.002833D0+ .004553D0*humid)/tamb  ! rhoa
        den(4,k) = 0.0D0                ! enthalpy of ambient air; TR=tamb
        den(5,k) = tamb
c
```

```
c
      do 300 i= ils,1,-1
      zbda = (float(i)/float(iils)) / (1.+humid)
      zw = zbda*humid
      zg = 1.D0-zbda-zw
c
c     enmix = zg*enthalpy + zbda*cpa*(tamb-tamb) + zw*cpw*(tamb-tamb) ! TR=tamb
      enmix = zg*enthalpy
c
      zbda = zbda + zg*wa
      zg   =        zg*wc
      call tprop(2,zg,zbda,enmix,yc,ya,wm,temp,rho,cp)
      cc = zg*rho
c
c
      curnt(1) = yc
      curnt(2) = cc
      curnt(3) = rho
      curnt(4) = enmix
      curnt(5) = temp
c
      if(i .eq. ils) then
            ind = 1
            do 150 jj= 1,5
150         backsp(jj,ind) = curnt(jj)
            goto 300
            endif
c
c     ADIABAT interpolation scheme
c
      err = 0.
      do 180 iind = 1,ind
      yc    = backsp(1,iind)
      cc    = backsp(2,iind)
      rho   = backsp(3,iind)
      enmix = backsp(4,iind)
      temp  = backsp(5,iind)
      slope = (den(2,k)- curnt(2)) / (den(1,k)- curnt(1))
            ccint = (yc - curnt(1))*slope + curnt(2)
      err = dmax1(err,2.D0* abs(cc - ccint)/(abs(cc + ccint) + zero))
      slope = (den(3,k)- curnt(3)) / (den(1,k)- curnt(1))
            rhoint = (yc - curnt(1))*slope + curnt(3)
      err = dmax1(err,2.D0* abs(rho - rhoint)/(abs(rho + rhoint) + zero))
      wccal = cc / rhoint
            w1 = curnt(2)/curnt(3)
            w2 = den(2,k)/den(3,k)
      slope = (den(4,k)- curnt(4)) / (w2 - w1)
            entint = (wccal - w1)*slope + curnt(4)
      err = dmax1(err,2.D0* abs(enmix - entint)/(abs(enmix + entint) + zero))
      slope = (den(5,k) - curnt(5)) / (w2 - w1)
            temint = (wccal - w1)*slope + curnt(5)
```

```
        err = dmax1(err,2.D0* abs(temp - temint)/(abs(temp + temint) + zero))
 180    continue
c
        if(err .le. tcrit) then
                if(ind .ge. iback) goto 200
                ind = ind + 1
                do 190 jj=1,5
 190            backsp(jj,ind) = curnt(jj)
                goto 300
                endif
c
c       record a point in DEN
c
c
 200    k = k+1
        if(k.ge. igen) call trap(28)
        do 250 jj=1,5
        den(jj,k) = backsp(jj,ind)
 250    backsp(jj,1) = curnt(jj)
        ind = 1
c
 300    continue
c
        k = k+1
        if(k.ge. igen) call trap(28)
        if(wc.eq. 1.D00) then
                den(1,k) = 1.D00        ! yc
                den(2,k) = gas_rhoe     ! cc
                den(3,k) = gas_rhoe     ! rhoe
                den(4,k) = enthalpy     ! enthalpy
                den(5,k) = gas_temp     ! temp
        else
        call tprop(2,wc,wa,enthalpy,den(1,k),ya,wm,den(5,k),den(3,k),cp)
        den(2,k) = wc*den(3,k)  ! cc
        den(4,k) = enthalpy
        endif
        den(1,k+1) = 2.         ! .gt. 1. end-of-record indicator
c
        return
        end
c
c
c
c
        subroutine addheat(cc,dh,rho,temp,cp)
c

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADIS1.dec'
c
```

```
        parameter (acrit=0.02D0)
c
        common
     $/GEN2/ DEN(5,igen)
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     ./ctprop/ cwc,cwa,centh
c
        character*3 gas_name

        external enthal0
c
c*** data for air/water sys
c
        data wma/28.96D0/       ! molecular weight of air
        data wmw/18.02D0/       ! molecular weight of water
        data rho_water/1000.D0/ ! liquid water density [=] kg/m**3
        data cpa/1.0063D3/      ! heat capacity of air [=]J/kg/K
        data cpw/1865.D0/       ! heat capacity of water vapor[=]J/kg/K
        data dhvap/2.5023D6/    !latent heat of vap [=]J/kg water
        data dhfus/0.33D6/      !latent heat of fus [=]J/kg water

        data rgas/0.08205D0/    ! gas constant for P[=]atm, T[=]K
c
c
c
        vapor_p(txxx) = 6.0298D-3* exp(5407.D0 *(1.D0/273.15D0- 1.D0/txxx))
c
c
        cp = cpa
        rhoa = den(3,1)
c
        call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enthalpy,amt)
        ww = 1.D0 - wc - wa
        yw = 1.D0 - yc - ya
        temp = amt
        IF(isofl.eq.1 .or. ihtfl.eq.0) return   ! adiabatic mixing is valid
        if(dh.le. 0.) return               ! catch colder surface temperatures
        enthalpy = enthalpy + dh
c
c
        if(enthalpy.gt. 0.) then
               temp = tamb
               goto 400
               endif
c
c
c
        tmin = amt      ! adiabatic mixing temp
        tmax = dmax1(gas_temp, tsurf, tamb)
        ehi = enthal(wc, wa, tmax)
        if(enthalpy.gt. ehi) then
            temp = tmax
            goto 400
        endif

        cwc = wc
        cwa = wa
        centh = enthalpy

c

        call zbrent(temp, enthal0, tmin, tmax, acrit, ierr)
c
c... If there are troubles, get LIMIT to help
c
        if(ierr .ne. 0) then
                tinc = (tmin+tmax)/100.D0
                temp = tmin
                tmin = tmin/2.D0
                tmax = 2.D0*tmax
                call limit(enthal0, temp, tinc, tmax, tmin)
                call zbrent(temp, enthal0, tmin,tmax,acrit,ierr)
                   if(ierr .eq. 0) goto 400
                call trap(17)
        endif
c
c
 400    continue                        ! density calculation
        vp = vapor_p(temp)
        ywsat = dmax1( vp/pamb, 0.D0)
        wwsat = wmw/wm * ywsat * (1.D0 - (yw-ywsat))
        conden = dmax1( 0.D0, ww-wwsat)

        rho = 1.D0/(      wa / (pamb*wma/rgas/temp)
     *                  + (ww-conden) / (pamb*wmw/rgas/temp)
     *                  + conden / rho_water
     *                  + wc * temp/gas_temp/gas_rhoe )

        if(temp.ne.amt) cp = dmax1(dh/(temp-amt),cpa)
c
c
c
        return
        end
####
```

```
C..............................................................
C
C       FILE NAME TRANS1 -- FOR USE IN DEGADIS1
C
C..............................................................
C
        SUBROUTINE TRANS(FILE)
C

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADIS1.dec'
c
C       BLOCK COMMON
C
        COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/TITL/TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/T1,TINP,TSRC,TOBS,TSRT
     $/ERROR/STPIN,ERBND,STPMX,WTRG,WTtm,WTya,wtyc,wteb,wtmb,wtuh,XLI,
     $ XRI,EPS,ZLOW,STPINZ,ERBNDZ,STPMXZ,SRCOER,srcss,srccut,
     $ htcut,ERNOBL,NOBLpt,crfger,epsilon
     $/PARM/UO,ZO,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl,swcl,swal,senl,srhl
     $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/com_enthal/ h_masrte,H_airrte,H_watrte
     $/NEND/ POUNDN,POUND
     $/ALP/ ALPHA,alpha1
     $/phicom/ iphifl,dellay
     $/sprd_con/ ce, delrhomin
     $/COM_SURF/ HTCUTS
     ./oomsin/ oodist,avtime
C
        character*80 TITLE(4)
C
        character*4 pound
        character*24 TSRC,TINP,TOBS,TSRT
        character*3 gas_name
C
        REAL*8 ML,K
        LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
```

```
c
        character*(*) file
c
        OPEN(UNIT=8,NAME=FILE,TYPE='NEW',
     $  carriagecontrol='list',
     $  recordtype='variable')
c
        WRITE(8,1000) (TITLE(I),I=1,4)
 1000   FORMAT(A80)
c
        DO 100 I=1,igen
  100   IF(PTIME(I).EQ.POUNDN) GO TO 105
        write(6,*) ' POUND WAS NOT DETECTED '
  105   NP = I - 1
        WRITE(8,1040) NP
        DO 110 I=1,NP
  110   WRITE(8,1030) PTIME(I),ET(I),R1T(I), PWC(I),PTEMP(I),
     $      PFRACV(I), PENTH(I), PRHO(I)
c
        DO 120 I=1,igen
  120   IF(DEN(1,I) .gt. 1.) GOTO 125
        DO 122 I=1,igen
  122   WRITE(8,1060) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
        write(6,*) ' density function blew the loop'
  125   NP = I - 1
        WRITE(8,1040) NP
        DO 130 I=1,NP
  130   WRITE(8,1060) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
c
        DO 140 I=1,maxl
c       cc = srcwc(2,i)*srcden(2,i)
c       if(cc.lt. cclow) then
c             fcc = 0.
c             do ii=i+1,maxl
c             fcc = amax1(srcwc(2,ii)*srcden(2,ii),fcc)
c             enddo
c             if(fcc.ge. cc) goto 140
c             np = i
c             tend = srcwc(1,i)
c             goto 146
c             endif
  140   IF(radg(1,I).EQ.POUNDN .AND. radg(2,I).EQ.POUNDN) GO TO 145
        write(6,*) ' POUND WAS NOT DETECTED '
  145   NP = I - 1
  146   WRITE(8,1040) NP
        DO 150 I=1,NP
  150   WRITE(8,1060) radg(1,i),radg(2,i),qstr(2,i),srcden(2,i),srcwc(2,i)
        1       ,srcwa(2,i),srcenth(2,i)
c
 1020   format(1x,i4,1x,1pg14.7)
 1030   format(8(1x,1pg14.7))
 1040   format(1x,i4)
 1050   format(2(a24,1x))
 1060   format(1x,1pg23.16,7(1x,1pg14.7))
 1070   format(1x,1pg14.7)
 1080   format(a3)
c
        WRITE(8,1050) TINP,TSRC
        write(8,1050) TOBS,TSRT
        write(8,1030) oodist,avtime
        WRITE(8,1060) U0,Z0,ZR,ML,USTAR
        write(8,1060) K,G,RHOE,RHOA,DELTA
        write(8,1030) BETA,GAMMAF,CcLOW
        WRITE(8,1060) RM,SZM,EMAX,RMAX,TSC1
        write(8,1030) ALEPH,TEND
        WRITE(8,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
        WRITE(8,1070) ALPHA
        write(8,1080) gas_name
        write(8,1030) gas_mw,gas_temp,gas_rhoe
        write(8,1030) gas_cpk,gas_cpp
        write(8,1030) gas_ufl,gas_lfl,gas_zsp
        write(8,1040) istab
        write(8,1030) tamb,pamb,humid
        write(8,1020) isofl,tsurf
        write(8,1020) ihtfl,htco
        write(8,1020) iwtfl,wtco
        write(8,1030) sigx_coeff,sigx_pow,sigx_min_dist
C
        if(check4) then
                write(8,1030) ess,slen,swid
                write(8,1060) outcc,outsz,outb,outl
                write(8,1060) swcl,swal,senl,srhl
                end if
c
        write(8,1020) iphifl,dellay
c
        if(isofl.eq. 0) write(8,1030) H_masrte
c
        WRITE(8,1030) HTCUTS, ce, delrhomin
C
        CLOSE(UNIT=8)
C
        RETURN
        END
####
```

```
C.............................................................
C
C       FILE NAME TRANS2 -- USE WITH DEGADIS2
C
        SUBROUTINE TRANS(FILE)
C

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADIS2.dec'
c
        COMMON
     $/SSCON/ NREC(maxnob,2),TO(maxnob),XV(maxnob)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/PARMSC/ RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/nend/ poundn,pound
     $/ALP/ ALPHA,alpha1
     $/CNOBS/ NOBS
     ./oomsin/ oodist,avtime
c
        character*3 gas_name
        character*80 TITLE(4)
        character*24 TINP,TSRC,TOBS,TSRT
        character*(*) file
c
        REAL*8 K,ML
        LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
        OPEN(UNIT=9,NAME=FILE,TYPE='NEW',
     $  carriagecontrol='list',
     $  recordtype='variable')
C
        WRITE(9,1040) NOBS
        DO 125 I=1,NOBS
  125   WRITE(9,1010) NREC(I,1),NREC(I,2),TO(I),XV(I)
c
        DO 140 I=1,igen
  140   IF(DEN(1,I).gt. 1.) GOTO 145
        write(6,*) ' density function error in TRANS'
  145   NP = I - 1
        WRITE(9,1040) NP
        DO 150 I=1,NP
  150   WRITE(9,1060) DEN(1,I),DEN(2,I),den(3,i),den(4,i),den(5,i)
```

```
c
        write(9,1060) oodist,avtime
c
        WRITE(9,1060) U0,Z0,ZR,ML,USTAR
        write(9,1060) K,G,RHOE,RHOA,DELTA
        write(9,1030) BETA,GAMMAF,CcLOW
c
        WRITE(9,1050) TINP,TSRC
        write(9,1050) TOBS,TSRT
c
        WRITE(9,1060) RM,SZM,EMAX,RMAX,TSC1
        write(9,1020) ALEPH,TEND
c
        write(9,1080) gas_name
        write(9,1030) gas_mw,gas_temp,gas_rhoe
        write(9,1020) gas_cpk,gas_cpp
        write(9,1030) gas_ufl,gas_lfl,gas_zsp
        write(9,1040) istab
        write(9,1030) tamb,pamb,humid
        write(9,1025) isofl,tsurf
        write(9,1025) ihtfl,htco
        write(9,1025) iwtfl,wtco
        write(9,1030) sigx_coeff,sigx_pow,sigx_min_dist
c
        WRITE(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
c
        WRITE(9,1070) ALPHA
c
 1010   format(1x,i8,1x,i8,2(1x,1pg14.7))
 1020   format(2(1x,1pg14.7))
 1025   format(1x,i4,1x,1pg14.7)
 1030   format(3(1x,1pg14.7))
 1040   format(1x,i4)
 1050   format(2(a24,1x))
 1060   format(5(1x,1pg14.7))
 1070   format(1x,1pg14.7)
 1080   format(a3,1x)
C
        CLOSE(UNIT=9)
        RETURN
        END
####
```

```
C...............................................................
C
C       FILE NAME TRANS2 -- USE WITH SDEGADIS2
C
C...............................................................
C
        SUBROUTINE TRANS(FILE)
C
C
C
C

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        COMMON
     $/PARM/ U0,Z0,ZR,ML,USTAR,K,G,RHOE,RHOA,DELTA,BETA,GAMMAF,CcLOW
     $/com_gprop/ gas_mw,gas_temp,gas_rhoe,gas_cpk,gas_cpp,
     $ gas_ufl,gas_lfl,gas_zsp,gas_name
     $/comatm/ istab,tamb,pamb,humid,isofl,tsurf,ihtfl,htco,iwtfl,wtco,
     $ humsrc
     $/ITI/ t1,TINP,TSRC,TOBS
     $/PHLAG/CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/ALP/ALPHA,alpha1
C
        character*24 TSRC,TINP,TOBS
        character*3 gas_name
        character*(*) file
C
        REAL*8 K,ML
        LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
C
        OPEN(UNIT=9,NAME=FILE,TYPE='NEW')
C
        WRITE(9,1060) U0,Z0,ZR,ML,USTAR
        write(9,1060) K,G,RHOE,RHOA,DELTA
        write(9,1030) BETA,GAMMAF,CcLOW
c
        WRITE(9,1050) TINP,TSRC
        write(9,1050) TOBS
c
        write(9,1080) gas_name
        write(9,1030) gas_mw,gas_temp,gas_rhoe
        write(9,1020) gas_cpk,gas_cpp
        write(9,1030) gas_ufl,gas_lfl,gas_zsp
c
        write(9,1040) istab
        write(9,1030) tamb,pamb,humid
        write(9,1025) isofl,tsurf
        write(9,1025) ihtfl,htco
        write(9,1025) iwtfl,wtco
c
```

```
       WRITE(9,*) CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
       WRITE(9,1070) ALPHA
 C
       CLOSE(UNIT=9)
 c
  1020 format(2(1x,1pg14.7))
  1025 format(1x,i4,1x,1pg14.7)
  1030 format(3(1x,1pg14.7))
  1040 format(1x,i4)
  1050 format(2(a24,1x))
  1060 format(5(1x,1pg14.7))
  1070 format(1x,1pg14.7)
  1080 format(a3,1x)
 C
       RETURN
       END
####
```

```
C       FILE NAME TRANS3 FOR USE WITH DEGADIS3
C
C............................................................
C
        SUBROUTINE TRANS(OPNRUP)
C

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADIS3.dec/list'
C
        COMMON /SORT/TCc(maxnob,maxnt),TCcSTR(maxnob,maxnt),
     $  Tyc(maxnob,maxnt),Trho(maxnob,maxnt),
     $  Tgamma(maxnob,maxnt),Ttemp(maxnob,maxnt),
     $  TSY(maxnob,maxnt),TSZ(maxnob,maxnt),TB(maxnob,maxnt),
     $  TDISTO(maxnob,maxnt),TDIST(maxnob,maxnt),KSUB(maxnt)
     $/SORTIN/TIM(maxnt),NTIM,ISTRT
     $/ITI/T1,TINP,TSRC,TOBS,TSRT
C
        character*24 tinp,tsrc,tobs,tsrt
C
        character*(*) OPNRUP
C
        TO = TIM(ISTRT)
        DT = TIM(ISTRT+1) - TIM(ISTRT)
C
        OPEN(UNIT=9,NAME=OPNRUP,TYPE='NEW',
     $  carriagecontrol='list',recordtype='variable')
C
C
        DO 110 I = ISTRT,NTIM
        II = I - ISTRT + 1
  110   KSUB(II) = KSUB(I)
        NTIM = NTIM - ISTRT + 1
        IF(NTIM .EQ. maxnt) GO TO 120
        II = NTIM + 1
        DO 115 I=II,maxnt
  115   KSUB(I) = 0
C
  120   CONTINUE
C
        WRITE(9,*) TO,DT,NTIM
        WRITE(9,*) KSUB
C
        CLOSE(UNIT=9)
C
        RETURN
        END
####
```

```
C.............................................................
C
C       SUBROUTINE TRAP -- DIAGNOSTICS
C
        SUBROUTINE trap(N,N1)
C

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADIS2.dec'
c
        COMMON /ITI/T1,TINP,TSRC,TOBS,TSRT

        real*4 tt1
c
        character*24 TINP,TSRC,TOBS,TSRT
c
        character*24 tt
        character*80 dd
C
        WRITE(lunlog,1100)
        WRITE(lunlog,1110)
        write(lunlog,1115) n
c
c*** check to see if the operator is ready to read the text of the error
c*** message
c
        write(lunlog,1000)
        irtn = lib$get_command( dd )    ! get a line from the terminal
 10     write(lunlog,1002)
C
        IF(N .EQ. 1 ) then
                WRITE(lunlog,2010) N1
                WRITE(lunlog,2011)
        else IF(N .EQ. 2 ) then
                WRITE(lunlog,2020)
        else IF(N .EQ. 3 ) then
                WRITE(lunlog,2030) N1
        else IF(N .EQ. 4 ) then
                WRITE(lunlog,2040)
        else IF(N .EQ. 5 ) then
                WRITE(lunlog,2050)
        else IF(N .EQ. 6 ) then
                WRITE(lunlog,2060)
        else IF(N .EQ. 7 ) then
                WRITE(lunlog,2070)
        else IF(N .EQ. 8 ) then
                WRITE(lunlog,2080) N1
                WRITE(lunlog,2081)
        else IF(N .EQ. 9 ) then
                WRITE(lunlog,2090) N1
```

```
                        WRITE(lunlog,2091)
        else IF(N .EQ. 10) then
                        WRITE(lunlog,2100) N1
                        WRITE(lunlog,2101)
        endif
        IF(N .EQ. 11) WRITE(lunlog,2110)
        IF(N .EQ. 12) WRITE(lunlog,2120)
        IF(N .EQ. 13) WRITE(lunlog,2130)
        IF(N .EQ. 14) WRITE(lunlog,2140)
        IF(N .EQ. 15) WRITE(lunlog,2150)
        IF(N .EQ. 16) WRITE(lunlog,2160)
        IF(N .EQ. 17) WRITE(lunlog,2170)
        IF(N .EQ. 18) then
                        WRITE(lunlog,2180) N1
                        WRITE(lunlog,2181)
        endif
        IF(N .EQ. 19) WRITE(lunlog,2190) N1
        IF(N .EQ. 20) WRITE(lunlog,2200)
        IF(N .EQ. 21) WRITE(lunlog,2210)
        IF(N .EQ. 22) WRITE(lunlog,2220)
        IF(N .EQ. 23) then
                        WRITE(lunlog,2230) maxnob
                        WRITE(lunlog,2231)
        endif
        IF(N .EQ. 24) WRITE(lunlog,2240)
        IF(N .EQ. 25) WRITE(lunlog,2250)
        IF(N .EQ. 26) WRITE(lunlog,2260)
        IF(N .EQ. 27) WRITE(lunlog,2270)
        IF(N .EQ. 28) WRITE(lunlog,2280)
        IF(N .EQ. 29) WRITE(lunlog,2290)
        IF(N .EQ. 30) WRITE(lunlog,2300)
        IF(N .EQ. 31) WRITE(lunlog,2310)
        IF(N .EQ. 32) WRITE(lunlog,2320)
        IF(N .EQ. 33) WRITE(lunlog,2330)
C
 1000   format(/,5x,'Ready for the text of the error message? <CR> ',$)
 1002   format(/)
 1100   FORMAT(5X,'The best laid plans of mice and men...')
 1110   FORMAT(5X,'You have entered a TRAP -- the land of no RETURN.')
 1115   format(' Code: ',i4)

 2010   FORMAT(5X,'DEGADIS1? Source integration has returned IHLF=',I3,/,
    ./,'     This error occurs during integration of the equations',/,
     .' which describe the gas source.  IHLF is an error code',/,
     .' returned by the integration package RKGST.',//,
     .'     When IHLF=11, more than 10 bisections of the initial',/,
     .' increment of the independent variable were necessary to make',/,
     .' an integration step within the specified error.  Reduce the',/,
     .' initial step size of the independent variable',/,
     .' (STPIN in the ER1 file).  If this does not work,',/,
     .' it will be necessary to either increase the error criteria',/,
     .' for all of the dependent variables being integrated',/,
     .' (ERBND in the ER1 file) or increase the error criteria',/,
     .' for the variable violating the criteria by decreasing the',/,
     .' error weight for that variable (one of the following: WTRG,',/,
     .' WTTM, WTYA, WTYC, WTEB, WTMB, or WTUH in the ER1 file).',/)
2011    format(
     .'     When IHLF=12, the initial increment of the independent',/,
     .' variable (STPIN) is 0.  Correct the ER1 file and execute the',/,
     .' program again.',//,
     .'     When IHLF=13, the initial increment of the independent',/,
     .' variable (STPIN) is not the same sign as the difference',/,
     .' between the upper bound of the interval and the lower bound',/,
     .' of the interval.  STPIN must be positive.  Correct the ER1',/,
     .' file and execute the program again.',//)

2020    FORMAT(5X,'Reserved')

2030    format(5x,'SZF? Local integration failed; IHLF=',I3,/,
     ./,'     This error occurs during estimation of SZ over the',/,
     .' source when no gas is present.  IHLF is an error code',/,
     .' returned by the integration package RKGST.',//,
     .'     When IHLF=11, more than 10 bisections of the initial',/,
     .' increment of the independent variable were necessary to make',/,
     .' an integration step within the specified error.  Reduce the',/,
     .' initial step size of the independent variable',/,
     .' (SZSTP0 in the ER1 file).  If this does not work,',/,
     .' increase the error criteria for all of the dependent',/,
     .' variables being integrated (SZERR in the ER1 file).',//,
     .'     When IHLF=12, the initial increment of the independent',/,
     .' variable (SZSTP0) is 0.  Correct the ER1 file and execute the',/,
     .' program again.',//,
     .'     When IHLF=13, the initial increment of the independent',/,
     .' variable (SZSTP0) is not the same sign as the difference',/,
     .' between the upper bound of the interval and the lower bound',/,
     .' of the interval.  SZSTP0 must be positive.  Correct the ER1',/,
     .' file and execute the program again.',//)

2040    format(5x,'SURFACE? Negative QRTE for positive DELTA_T',//,
     .'     This is a diagnostic message indicating an error in',/,
     .' estimation of the heat capacity.  Check the input',/,
     .' to the model and execute the program again.',//)

2050    FORMAT(5X,'CRFG? MORE POINTS FOR GEN3 WERE NEEDED',//,
     .'     The COMMON area /GEN3/ stores representative values',/,
     .' of the calculated source parameters.  If this message',/,
     .' occurs, relax the CRFG error criteria (CRFGER) in the',/,
     .' ER1 file.  If this is a common problem, the length of the',/,
     .' /GEN3/ vectors can be increased by changing the value of',/,
     .' MAXL in DEGADIS1.DEC and reinstalling DEGADIS.',//)

2060    FORMAT(5X,'TUPF? OBSERVER CALCULATIONS -- TUPF FAILED',//,
     .'     The trial and error search associated with finding the',/,
     .' upwind edge of the gas source for an observer failed.',/,
     .' Often, this problem can be avoided by adding one or two',/,
     .' additional observers to the present number of observers',/,
     .' (which changes the conditions for the trial and error).',/,
     .' Another possibility is to increase the error criteria for',/,
     .' this function (ERTUPF) in the ER2 file.',//)

 2070   FORMAT(5X,'TUPF? OBSERVER CALCULATIONS -- TDNF FAILED',//,
     .'     The trial and error search associated with finding the',/,
     .' downwind edge of the gas source for an observer failed.',/,
     .' Often, this problem can be avoided by adding one or two',/,
     .' additional observers to the present number of observers',/,
     .' (which changes the conditions for the trial and error).',/,
     .' Another possibility is to increase the error criteria for',/,
     .' this function (ERTDNF) in the ER2 file.',//)

 2080   FORMAT(5X,'SSSUP? OBSERVER INTEGRATION FAILED, IHLF=',I3,//,
     .'     This error occurs during integration of the five',/,
     .' differential equations which average the source for each',/,
     .' observer.  IHLF is an error code returned by the',/,
     .' integration package RKGST.',//,
     .'     When IHLF=11, more than 10 bisections of the initial',/,
     .' increment of the independent variable were necessary to make',/,
     .' an integration step within the specified error.  Reduce the',/,
     .' initial step size of the independent variable',/,
     .' (STPO in the ER2 file).  If this does not work,',/,
     .' it will be necessary to either increase the error criteria',/,
     .' for all of the dependent variables being integrated',/,
     .' (ERRO in the ER2 file) or increase the error criteria',/,
     .' for the variable violating the criteria by decreasing the',/,
     .' error weight for that variable (one of the following: WTAIO,',/,
     .' WTQOO, or WTSZO in the ER2 file).',/)
 2081   format(
     .'     When IHLF=12, the initial increment of the independent',/,
     .' variable (STPO) is 0.  Correct the ER2 file and execute the',/,
     .' program again.',//,
     .'     When IHLF=13, the initial increment of the independent',/,
     .' variable (STPO) is not the same sign as the difference',/,
     .' between the upper bound of the interval and the lower bound',/,
     .' of the interval.  STPO must be positive.  Correct the ER2',/,
     .' file and execute the program again.',//)

 2090   FORMAT(5X,'SSSUP/SDEGADIS2? PSEUDO-STEADY INTEG FAILED, IHLF=',I3,
     .//,'     This error occurs during integration of the four',/,
     .' differential equations describing the portion of the',/,
     .' downwind calculation when b>0.  The routine calling TRAP is',/,
     .' SSSUP if a transient simulation is being executed; if a',/,
     .' steady state simulation is being executed, the calling',/,
     .' routine is SDEGADIS2.  IHLF is an error code returned by the',/,
     .' integration package RKGST.',//,
     .'     When IHLF=11, more than 10 bisections of the initial',/,
     .' increment of the independent variable were necessary to make',/,
     .' an integration step within the specified error.  Reduce the',/,
     .' initial step size of the independent variable',/,
     .' (STPP in the ER2 file).  If this does not work,',/,
     .' it will be necessary to either increase the error criteria',/,
     .' for all of the dependent variables being integrated',/,
     .' (ERRP in the ER2 file) or increase the error criteria',/,
     .' for the variable violating the criteria by decreasing the',/,
     .' error weight for that variable (one of the following: WTSZP,',/,
     .' WTSYP, WTBEP, or WTDH in the ER2 file).',/)
2091  format(
     .'     When IHLF=12, the initial increment of the independent',/,
     .' variable (STPP) is 0.  Correct the ER2 file and execute the',/,
     .' program again.',//,
     .'     When IHLF=13, the initial increment of the independent',/,
     .' variable (STPP) is not the same sign as the difference',/,
     .' between the upper bound of the interval and the lower bound',/,
     .' of the interval.  STPP must be positive.  Correct the ER2',/,
     .' file and execute the program again.',//)

2100  FORMAT(5X,'SSSUP/SDEGADIS2? GAUSSIAN INTEGRATION FAIL, IHLF=',I3,
     .//,'     This error occurs during integration of the',/,
     .' differential equations describing the portion of the',/,
     .' downwind calculation when b=0.  The routine calling TRAP is',/,
     .' SSSUP if a transient simulation is being executed; if a',/,
     .' steady state simulation is being executed, the calling',/,
     .' routine is SDEGADIS2.  IHLF is an error code returned by the',/,
     .' integration package RKGST.',//,
     .'     When IHLF=11, more than 10 bisections of the initial',/,
     .' increment of the independent variable were necessary to make',/,
     .' an integration step within the specified error.  Reduce the',/,
     .' initial step size of the independent variable',/,
     .' (STPG in the ER2 file).  If this does not work,',/,
     .' it will be necessary to either increase the error criteria',/,
     .' for all of the dependent variables being integrated',/,
     .' (ERRG in the ER2 file) or increase the error criteria',/,
     .' for the variable violating the criteria by decreasing the',/,
     .' error weight for that variable (either WTRUH or WTDHG',/,
     .' in the ER2 file).',//)
2101  format(
     .'     When IHLF=12, the initial increment of the independent',/,
     .' variable (STPG) is 0.  Correct the ER2 file and execute the',/,
     .' program again.',//,
     .'     When IHLF=13, the initial increment of the independent',/,
     .' variable (STPG) is not the same sign as the difference',/,
     .' between the upper bound of the interval and the lower bound',/,
     .' of the interval.  STPG must be positive.  Correct the ER2',/,
     .' file and execute the program again.',//)

2110  FORMAT(5X,'SSSUP/SDEGADIS2? TOTAL No. OF RECORDS EXCEED 120000',
     .//,'     This is an arbitrary stopping point for the process',/,
     .' in order to keep a runaway simulation from filling up disk',/,
     .' space.  Relax the output specifications (ODLP, ODLLP, ODLG,',/,
     .' or ODLLG) in the ER2 file in order to generate less output',/,
     .' if the input parameters are valid.',//)

2120  FORMAT(5X,'Reserved')
2130  FORMAT(5X,'Reserved')
2140  FORMAT(5X,'Reserved')
2150  FORMAT(5X,'Reserved')

2160  FORMAT(5X,'PSSOUT/PSSOUTSS? PSS STARTED WITH B<0.',//,
     .'     This condition is checked at the beginning of the',/,
     .' downwind calculation in order to confirm proper handling of',/,
     .' the movement to the Gaussian phase of the downwind ',/,
     .' calculation.  Check the initial conditions and execute the',/,
     .' program again.',//)

2170  format(5x,'TPROP/ADDHEAT? Enthalpy out of bounds',//,
     .'     Diagnostic message indicates an enthalpy lower',/,
     .' than the adiabatic mixing enthalpy was passed to ADDHEAT.',/,
     .' Check the input conditions and execute the program again.',//)

2180  FORMAT(5X,'ALPH? ALPHA INTEGRATION FAILED, IHLF=',I3,//,
     .'     The integration which determines the integral least',/,
     .' squares fit for ALPHA has failed.  Note that small values',/,
     .' of the Monin-Obukhov length ( ML < O(1m) ) in combination',/,
     .' with stable atmospheric conditions may cause this failure.',/,
     .' IHLF is an error code returned by the integration package',/,
     .' RKGST.',//,
     .'     When IHLF=11, more than 10 bisections of the initial',/,
     .' increment of the independent variable were necessary to make',/,
     .' an integration step within the specified error.  Reduce the',/,
     .' absolute value of the initial step size of the independent',/,
     .' variable (STPINZ in the ER1 file).  If this does not work,',/,
     .' it will be necessary to increase the error criteria',/,
     .' (ERBNDZ in the ER1 file).',//)
2181  format(
     .'     When IHLF=12, the initial increment of the independent',/,
     .' variable (STPINZ) is 0. Correct the ER1 file and execute',/,
     .' the program again.',//,
     .'     When IHLF=13, the initial increment of the independent',/,
     .' variable (STPINZ) is not the same sign as the difference',/,
     .' between the upper bound of the interval and the lower bound',/,
     .' of the interval.  STPINZ must be negative.  Correct the ER1',/,
     .' file and execute the program again.  This error will also',/,
     .' occur if the surface roughness ZR is greater than the',/,
     .' reference height ZO.',//)

2190  FORMAT(5X,'ALPH? ZBRENT has failed to locate ALPHA IERR: ',I4,//,
     .'     The search procedure which determines ALPHA has failed.',/,
     .' This error may be the result of an unusual velocity',/,
     .' specification such as small values of the Monin-Obukvov',/,
     .' length ( ML < O(1.m) ) or small reference heights',/,
     .' ( Z0 < O(10. * ML) ).  IERR is an error code returned by',/,
     .' the routine ZBRENT.',//,
     .'     When IERR=1, the search for ALPHA failed after a',/,
     .' specified number of iterations. Increase the error bound',/,
     .' used by ZBRENT (EPS in ER1 file).',//,
     .'     When IERR=2, the basic assumption that the function which',/,
     .' governs the search for ALPHA changes sign over the specified',/,
     .' interval is false.  Increase the search interval by',/,
     .' decreasing the lower bound of ALPHA (XLI in the ER1 file)',/,
     .' and increasing the upper bound (XRI in the ER1 file).',//)

2200  format(5x,'ESTRT? Premature EOF in RUN_NAME.ER1 or RUN_NAME.ER2.'
     .,//,'       The portion of the program which reads ER1 and',/,
     .' ER2 files encountered an end-of-file mark before all of',/,
     .' the information had been read.  Confirm these files and',/,
     .' execute the program again.  If necessary, copy and edit',/,
     .' the appropriate EXAMPLE file and execute the program again.',//)

2210  FORMAT(5X,'ESTRT1/ESTRT2/ESTRT2SS/ESTRT3? DECODE failed',//,
     .'     The portion of the program which reads the ER1, ER2,',/,
     .' or the ER3 file failed to understand a numerical entry.',/,
     .' The numbers must appear in columns 11-20 of the line with',/,
     .' no alphabetic characters in the field.  (Note that',/,
     .' exponential notation is not allowed.)  This restriction',/,
     .' does not apply to comment lines which have an exclamation',/,
     .' point (!) in the first column.',//)

2220  format(5x,'ESTRT1? The parameter file RUN_NAME.ER1 ',
     .'was not found.',//,
     .'     The ER1 file was not found for the current simualtion',/,
     .' (RUN_NAME).  Copy the file EXAMPLE.ER1 file to RUN_NAME.ER1',/,
     .' and edit it as necessary.  Execute the program again.',//)

2230  format(5x,'SORTS1? Fewer than 3 points sorted for any time.',//,
     .'     Only one or two simualtion points were applicable for',/,
     .' the sort times specified.  There are three possible causes',/,
     .' for this condition:',//,
     .'     (1) If this message appears when the sort times are',/,
     .' defaulted (CHECK5 is set to 0. in the ER3 file), the',/,
     .' number of observers will probably have to be increased',/,
     .' to give a good resolution of the downwind concentration',/,
     .' field.  (The number of observers is NOBS in the ER2 file',/,
     .' with a maximum of ',I3,' (MAXNOB) in DEGADIS2.DEC.)',/,
     .' As a rule of thumb, one gets good resolution of the downwind',/,
     .' concentration field if the ratio:',/,
     .' (secondary source duration / number of observers) is less',/,
     .' than about 10 seconds (or 20 at most).',//,
     .'     (2) The sort times specified in the ER3 file were',/,
     .' before the simulation had developed significantly.',/,
     .' This is only applicable when the user is specifying the sort',/,
     .' times (i.e. when CHECK5 is set to 1. in the ER3 file).',/,
     .' Increase the time of the first sort (ERT1), and rerun the',/,
     .' program.',/)
2231  format(
     .'     (3) The sort times specified in the ER3 file were',/,
     .' after the gas was below the lowest concentration of interest',/,
     .' This is only applicable when the user is specifying the sort',/,
     .' times (i.e. when CHECK5 is set to 1. in the ER3 file).',/,
     .' Increase the time of the first sort (ERT1), and rerun the',/,
     .' program.  If additional results are desired for later',/,
     .' times, restart the simulation and specify a lower',/,
     .' concentration of interest in the input step ( lower',/,
     .' CCLOW in DEGADISIN).',//,
     .'     (4) The downwind distance specified in DEGADIS4 is too',/,
     .' large for the lowest concentration of interest specified.  To',/,
     .' get a time history at this downwind distance, restart the',/,
     .' simulation with a lower concentration of interest in the input.',
     .//)

2240  format(5x,'TPROP? Trial and error loop compromised',//,
     .'     TPROP estimates the temperature of a mixture based',/,
     .' upon the composition and enthalpy of the mixture.  Ensure',/,
     .' the properties for the diffusing species are entered',/,
     .' correctly and execute the simulation again.',//)

2250  format(5x,'TPROP? Isothermal density loop compromised',//,
     .'     This error should never occur, but if it does, rebuild',/,
     .' the model from the original files and run the simulation',/,
     .' over.',//)

2260  format(5x,'TPROP? Invalid entry flag in ADIABAT',//,
     .'     This is a programming diagnostic and should never occur.',/,
     .' If it does, rebuild the model from the original files.',//)

2270  format(5x,'Reserved')

2280  format(5x,'TPROP? IGEN request too large in SETDEN',//,
     .'     The subroutine SETDEN (in TPROP) performs a series of',/,
     .' adiabatic mixing calculations with a specified gas mixture',/,
     .' and ambient air and places the result in the array',/,
     .' DEN(5,IGEN).  This error indicates more points are needed in',/,
     .' DEN than were originally requested.  Increase the allocation',/,
     .' for DEN by changing the value of IGEN in DEGADISIN.DEC and',/,
     .' reinstalling DEGADIS.',//)

2290  format(5x,'PHIF? flag IPHIFL is out of bounds',//,
     .'     Proper values of IPHIFL are integers between 1 and 5 ',/,
     .' inclusive.  Although values of IPHIFL are entered in the ER1',/,
     .' file as real numbers, they should be in this range.  Check',/,
     .' the ER1 file and execute the program again.',//)

 2300   format(5x,'SSSUP/SDEGADIS2? concentration greater than RHOE',//,
     .'     If the concentration of the contaminant becomes',/,
     .' greater than the pure component density for an isothermal',/,
     .' simulation, this error will occur.  However, this situation',/,
     .' should never occur.  Check the input conditions and execute',/,
     .' the program again.',//)

 2310   format(5x,'SSSUP? concentration greater than RHOE',//,
     .'     If the concentration of the contaminant becomes',/,
     .' greater than the pure component density for an isothermal',/,
     .' simulation, this error will occur.  However, this situation',/,
     .' should never occur.  Check the input conditions and execute',/,
     .' the program again.',//)

 2320   format(5x,'PSS? Sz convergence failure.',//,
     .'     This is a programming diagnostic and should never occur.',/,
     .' If it does, check the input conditions and execute the',/,
     .' program again.',//)

 2330   format(5x,'SSG? Sz convergence failure.',//,
     .'     This is a programming diagnostic and should never occur.',/,
     .' If it does, check the input conditions and execute the',/,
     .' program again.',//)
C
        CLOSE(UNIT=9)
C
c CALL TRANS('trap.DBG') - this isn't really needed
C
        istat = lib$date_time(TT)
        tt1 = t1
        ttime = secnds(tt1)/60.
C
  140   WRITE(lunlog,3000) TT
        WRITE(lunlog,3010) Ttime
 3000   FORMAT(1X,' -- ENDing AT ',A24)
 3010   FORMAT(5X,' ***** ELAPSED TIME ***** ',1pg13.5,' MIN ')
C
        irtn = LIBSDO_COMMAND( 'Exit' ) ! this issues the command EXIT to VMS
c                                       ! which should cancel any pending
c                                       ! programs in a command file.
        CALL EXIT
        END
####
```

```
C..........................................................
C
C       FUNCTION TO CALCULATE A SPECIFIED TIME
C
        FUNCTION TS(TOl,DIST)
C

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        COMMON
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
C
        TS = TOl + (DIST+RMAX)**(1./ALPHA1) /ALEPH
C
        RETURN
        END
###.
```

```
C.............................................................
C
C       OBSERVER TRIAL AND ERROR FUNCTIONS
C         --- TUPF ---- TDNF ---
c
c       Modified 30 Jan 86 for more general trial and error scheme. ts
C
        FUNCTION TUPF(TOL)
C
        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADIS2.dec'
c
        COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/ERROR/SYOER,ERRO,SZOER,WTAIO,WTQOO,WTSZO,ERRP,SMXP,
     $ WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WTDHG
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
C
        LOGICAL pflag           ! for diagnostic output
        pflag = .false.
C
        TMAX = RMAX**(1./ALPHA1)/ALEPH + TOL
        TMIN = TOL
        IF(TOL .LT. 0.) TMIN = 0.
c
c*** TMIN and TMAX represent the first and last time this observer could
c*** encounter the upwind edge of the source.  TMAX is the time when the
c*** observer passes over x=0, and TMIN is the time the observer is
c*** released (unless set to zero because the spill has not yet begun).
c*** Now, refine the guess of TMIN and TMAX by dividing the interval
c*** into 20 segments and checking if the observer crosses the upwind
c*** edge over the smaller interval starting with TMIN.
c
        DT = (TMAX - TMIN) / 20.
        TL = TMIN

        DO 10 I=1,19    ! I don't have to check the last interval.
        TL = TL + DT
        XO = XIT(TL, TOL)
        XG = -AFGEN(RADG, TL, 'tupf')
        DIF = XO - XG
        IF(DIF .LT. 0.) GOTO 10 ! observer has not yet reached upwind edge
              TMAX = TL       ! now observer has reached upwind edge
              TMIN = TL - DT
              DIFATMAX = DIF  ! DIF at MAX
              GOTO 20
10      CONTINUE
        TMIN = TMAX - DT
```

```
      DIFATMAX = DIF
   20 CONTINUE
c
c*** Now perform bisection search to get desired convergence between new TMAX
c*** and TMIN
c
      TL = (TMAX + TMIN)/2.
C
      DO 100 I = 1,20
      II = 0
  110 XG = -AFGEN(RADG,TL,'tupf')
      XO = XIT(TL,TOL)
      IF(XO .LT. 0.) GO TO 120
      TL = (TL+TMIN)/2.
      II = II + 1
      if(pflag) write(6,5020) tl,tOl,xg,xo
 5020 format(' tl:',1pg13.5,' tOl:',1pg13.5,' xg:',1pg13.5,
     1 ' xo:',1pg13.5)
      IF(II.EQ. 20) GOTO 101  ! Kill the program.
      GO TO 110
C
  120 CONTINUE
      DIF = XO - XG
      sum = abs(xo + xg)/2. + ERTUPF
      IF(ABS(DIF)/sum .LT. ERTUPF) GO TO 1000
      if(pflag) write(6,5040) tmin,tmax,tl,xo,xg
 5040 format(' tmin:',1pg13.5,' tmax:',1pg13.5,' tl:',1pg13.5,
     1 ' xo:',1pg13.5,' xg:',1pg13.5)
C
      IF( DIF*DIFATMAX .GT. 0.) THEN
            TMAX = TL                   ! a new maximum for the range
            DIFATMAX = DIF              ! a new DIF for the maximum
      ELSE
            TMIN = TL                   ! a new minimum for the range
      ENDIF
C
  100 TL = (TMAX + TMIN)/2.
C
c*** The above search scheme failed.  Before killing the program, check to
c*** see if the desired point falls on a transition from a blanket to a
c*** non-blanket situation.
c
      t1 = TL+.01
      T2 = TL-.01
      XG1 = AFGEN(RADG,T1,'tupf')
      xg2 = AFGEN(RADG,T2,'tupf')
      dif = abs(xg1-xg2)
      if(dif.gt. 100. .AND. (XO.GE.XG2 .AND. XO.LE.XG1)) then
            tupf = t2         ! jump from blanket to non-blanket occured
            RETURN
            ENDIF
```

```
c
c*** Kill the program.
c
 101    if(pflag) write(6,4000) RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND,alpha
 4000   format(' rm:',1pg13.5,' szm:',1pg13.5,' emax: ',1pg13.5,/,
        1 ' rmax:',1pg13.5,' tsc1:',1pg13.5,' aleph:',1pg13.5,/,
        2 ' tend:',1pg13.5,' alpha:',1pg13.5)
        if(pflag) write(6,4010) tmax,tmin
 4010   format(' tmax: ',1pg13.5,' tmin: ',1pg13.5)
              CALL trap(6)
C
c*** successful completion
c
 1000   TUPF = TL
        RETURN
        END
C
c
c
c
        FUNCTION TDNF(TOL)
C


        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        include 'sys$degadis:DEGADIS2.dec'
c
        COMMON
     $/GEN3/ radg(2,maxl),qstr(2,maxl),srcden(2,maxl),srcwc(2,maxl),
     $ srcwa(2,maxl),srcenth(2,maxl)
     $/ERROR/SYOER,ERRO,SZOER,WTAIO,WTQOO,WTSZO,ERRP,SMXP,
     $ WTSZP,WTSYP,WTBEP,WTDH,ERRG,SMXG,ERTDNF,ERTUPF,WTRUH,WTDHG
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
C
        LOGICAL pflag
        pflag = .FALSE.
C
        TMIN = RMAX**(1./ALPHA1)/ALEPH + TOL
        IF(TMIN .LT. 0.) TMIN = 0.
        TMAX = (2.*RMAX)**(1./ALPHA1)/ALEPH + TOL
c
c*** TMIN and TMAX represent the first and last time this observer could
c*** encounter the downwind edge of the source.  TMAX is the time when the
c*** observer passes over x=+RMAX, and TMIN is the time the observer
c*** passes over x=0 (unless set to zero because the spill has not begun
c*** yet).  Now, refine the guess of TMIN and TMAX by dividing the interval
c*** into 20 segments and checking if the observer crosses the downwind
c*** edge over the smaller interval starting from TMAX.
c
        DT = (TMAX - TMIN) / 20.
```

```
      TL = TMAX
      if(pflag) write(6,*) 'tmax, tmin, dt: ',tmax, tmin, dt

      DO 10 I=1,19     ! I don't have to check the last interval.
      TL = TL - DT
      XO = XIT(TL, TOL)
      XG = AFGEN(RADG, TL, 'tupf')
      DIF = XO - XG
      if(pflag) write(6,*) 'tl, xo, xg, dif:',tl, xo, xg, dif
      IF(DIF .GT. 0.) GOTO 10 ! observer has passed the downwind edge
              TMIN = TL       ! now observer is about to reach downwind edge
              TMAX = TL + DT
              GOTO 20
 10   CONTINUE
c
      TMAX = TMIN + DT
 20   CONTINUE
      XO = XIT(TMAX, TOL)
      XG = AFGEN(RADG, TMAX, 'tupf')
      DIFATMAX = XO - XG
      if(pflag) then
              write(6,*) '20: tmax, tmin: ',tmax, tmin
              write(6,*) 'difatmax: ',difatmax
      endif
C
c*** Now perform bisection search to get desired convergence between new TMAX
c*** and TMIN.
c
      TL = (TMAX + TMIN)/2.
C
      DO 100 I = 1,20
      II = 0
 110  XG = AFGEN(RADG, TL, 'tdnf')
      XO = XIT(TL,TOL)
C
      IF(XO .GT. 0.) GO TO 120
      TL = (TMAX + TL)/2.
      II = II + 1
      if(pflag) write(6,5020) tl,tOl,xg,xo
 5020 format(' tl:',1pg13.5,' tOl:',1pg13.5,' xg:',1pg13.5,
      1 ' xo:',1pg13.5)
      IF(II.EQ. 20) GOTO 101  ! Kill the program.
      GO TO 110
C
 120  CONTINUE
C
      DIF = XO - XG
      sum = abs(xo+xg)/2. + ERTDNF
      IF(ABS(DIF)/sum .LT. ERTDNF) GO TO 1000
      if(pflag) write(6,5040) tmin,tmax,tl,xo,xg
 5040 format(' tmin:',1pg13.5,' tmax:',1pg13.5,' tl:',1pg13.5,
     1 ' xo:',1pg13.5,' xg:',1pg13.5)
c
        IF( DIF*DIFATMAX .GT. 0.) THEN
                TMAX = TL                       ! a new maximum for the range
                DIFATMAX = DIF                  ! a new DIF for the maximum
        ELSE
                TMIN = TL                       ! a new minimum for the range
        ENDIF
c
  100   TL = (TMAX + TMIN)/2.
c
c*** The above search scheme failed.  Before killing the program, check to
c*** see if the desired point falls on a transition from a blanket to a
c*** non-blanket situation.
c
        t1 = TL+.01
        T2 = TL-.01
        XG1 = AFGEN(RADG,T1,'tupf')
        xg2 = AFGEN(RADG,T2,'tupf')
        dif = abs(xg1-xg2)
        if(dif.gt. 100. .AND. (XO.LE.XG2 .AND. XO.GE.XG1)) then
                tdnf = t2               ! jump from blanket to non-blanket occured
                RETURN
                ENDIF
c
c*** Kill the program.
c
  101   if(pflag) write(6,4000) RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND,alpha
 4000   format(' rm:',1pg13.5,' szm:',1pg13.5,' emax: ',1pg13.5,/,
     1 ' rmax:',1pg13.5,' tsc1:',1pg13.5,' aleph:',1pg13.5,/,
     2 ' tend:',1pg13.5,' alpha:',1pg13.5)
        if(pflag) write(6,4010) tmax,tmin
 4010   format(' tmax: ',1pg13.5,' tmin: ',1pg13.5)
                CALL trap(7)
c
c*** successful completion
c
 1000   TDNF = TL
        RETURN
        END
####
```

```
C............................................................
C
C       FUNCTIONS ASSOCIATED WITH THE OBSERVER CALCULATIONS
C
C............................................................
C
C       FUNCTION TO RETURN OBSERVER VELOCITY AS A FUNCTION OF TIME
C
        FUNCTION UIT(T,TOl)
C

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        COMMON
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
C
        UIT = ALPHA1 * ALEPH**ALPHA1 *(T-TOl)**ALPHA
C
        RETURN
        END
c
c
C..................................................................
C
C       FUNCTION TO RETURN POSITION AS A FUNCTION OF TIME AND TO
C
C
        FUNCTION XIT(Tl,TOl)
C

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

        COMMON
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
C
        xit = -rmax
        arg = tl-t0l
        if(arg .le. 0.) return
        XIT = (ALEPH*(Tl - TOl))**ALPHA1 - RMAX
C
        RETURN
        END
c
c
c..................................................................
c
C
C       FUNC TO RETURN A VALUE OF TO BASED ON A POSITION AND TIME
C
```

```
      FUNCTION TOOB(X,T)
C
      Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )
C
      COMMON
     $/PARMSC/RM,SZM,EMAX,RMAX,TSC1,ALEPH,TEND
     $/ALP/ALPHA,alpha1
C
      ARG = 0.
      CHECK = ABS((ABS(X)-ABS(RMAX)))/(ABS(X)+ABS(RMAX))
      IF(CHECK .GT. 0.001) ARG = (X + RMAX)**(1./ALPHA1)/ALEPH
      TOOB = T - ARG
      RETURN
      END
####
```

```
           answer = bbb
           return
        endif
c
c... attempt inverse quadratic interpolation
c
        if(abs(eee).ge.tol1 .and. abs(fa).gt.abs(fb)) then
           sss = fb/fa

           if(aaa.eq.ccc) then
               ppp = 2.D0*xm*sss
               qqq = 1.D0-sss
           else
               qqq = fa/fc
               rrr = fb/fc
               ppp = sss*(2.D0*xm*qqq*(qqq-rrr) - (bbb-aaa)*(rrr-1.D0))
                              qqq = (qqq-1.D0)*(rrr-1.D0)*(sss-1.D0)
           endif
c
c.......... check to insure in bounds
c
           if(ppp .gt. 0.D0) qqq = -qqq
               ppp = abs(ppp)
               if
    .  (2.D0*ppp .lt. min(3.D0*xm*qqq-abs(tol1*qqq),abs(eee*qqq))) then

c................. interpolation valid
                  eee = ddd
                  ddd = ppp/qqq
              else

c................. interpolation failed, use bisection
                  ddd = xm
                  eee = ddd
              endif
          else

c.............. bounds decreasing too slowly, use bisection.
              ddd = xm
              eee = ddd
          endif
c
c...... move last best guess to A
c
      aaa = bbb
      fa = fb
c
c... evaluate new trial root
c
      if(abs(ddd) .gt. tol1) then
              bbb = bbb+ddd
        else
                bbb = bbb+sign(tol1,xm)
        endif
        fb = func(bbb)
  11    continue
c
c... loop failed to converge the x range
c
        ier = 1
        return
        end
####
```

APPENDIX E

JETPLU-DEGADIS INTERFACE SOURCE CODE

(DEGBRIDGE)

(This page is intentionally left blank.)

```
      PROGRAM DEGBRIDGE
C
C-----------------------------------------------------------------------
C
C     Program description:
C
C     DEGBRIDGE is designed to read the input file for the OOMS/DEGADIS
C     model input.  From this information and the output from the OOMS model,
C     DEGBRIDGE prepares the necessary input file to run DEGADIS
C     (RUN_NAME.INP).
C
C-----------------------------------------------------------------------
C
C     The structure of the RUN_NAME.IN file is outlined in OOMS_IN.
C
C     The output file from the OOMS model contains three (3) variables
C     as follows (in RUN_NAME.IND):
C
C     <DIST>  <CC>  <HWIDTH>
C
C     where:
C
C       <DIST>   is the distance to the centerline touchdown (m)
C       <CC>     is the centerline concentration at <DIST>
C       <HWIDTH> is jet/plume half width at <DIST>
C
C     To establish the initial conditions for DEGADIS, the source
C     concentration is assumed to be <CC>, and the source radius is assumed
C     to be <HWIDTH>.
C
C-----------------------------------------------------------------------
C
C     Program usage:
C
C     Consult Volume III of the Final Report to U. S. Coast Guard
C     contract DT-CG-23-80-C-20029 entitled "Development of an
C     Atmospheric Dispersion Model for Heavier-than-Air Gas Mixtures".
C
C     J. A. Havens
C     T. O. Spicer
C
C     University of Arkansas
C     Department of Chemical Engineering
C     Fayetteville, AR 72701
C
C     April 1985
C
C
C     This project was sponsored by the U. S. Coast Guard and the Gas
C     Research Institute under contract DT-CG-23-80-C-20029.
C
```

```
C
C       Disclaimer:
C
C       This computer code material was prepared by the University of
C       Arkansas as an account of work sponsored by the U. S. Coast Guard
C       and the Gas Research Institute.  Neither the University of Arkansas,
C       nor any person acting on its behalf:
C
C       a.  Makes any warranty or representation, express or implied,
C           with respect to the accuracy, completeness, or usefulness
C           of the information contained in this computer code material,
C           or that the use of any apparatus, method, numerical model,
C           or process disclosed in this computer code material may not
C           infringe privately owned rights; or
C
C       b.  Assumes any liability with respect to the use of, or for
C           damages resulting from the use of, any information,
C           apparatus, method, or process disclosed in this computer
C           code material.
C
C
C*****************************************************************************
C*****************************************************************************
C*****************************************************************************
C
C
c

        Implicit Real*8 ( A-H, O-Z ), Integer*4 ( I-N )

C
        include 'SYS$DEGADIS:degadisin.dec'
c
        COMMON
     $/TITL/ TITLE
     $/GEN1/ PTIME(igen), ET(igen), R1T(igen), PWC(igen), PTEMP(igen),
     $       PFRACV(igen), PENTH(igen), PRHO(igen)
     $/GEN2/ DEN(5,igen)
     $/ITI/ T1,TINP,TSRC,TOBS,TSRT
     $/PARM/ U0,Z0,ZR,rml,USTAR,vkc,gg,gRHOE,RHOA,DELTA,BETA,GAMMAF,
     .          CcLOW
     $/com_gprop/ gasmw,temjet,rhoe,cpk,cpp,gasul,gasll,zll,gasnam
     $/com_ss/ ess,slen,swid,outcc,outsz,outb,outl
     $/PHLAG/ CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
     $/com_sigx/ sigx_coeff,sigx_pow,sigx_min_dist,sigx_flag
     $/NEND/ POUNDN,POUND
     ./oomsin/ oodist,avtime
C
       character*80 TITLE(4)
C
       character*3 gasnam
```

```
      character*4 pound
      character*24 TSRC,TINP,TOBS,TSRT
C
      LOGICAL CHECK1,CHECK2,AGAIN,CHECK3,CHECK4,CHECK5
c
c check1
c check2=t      cloud type release with no liquid source; SRC1   DEGADIS1
c again         local communications in SSSUP                    SSSUP
c check3        local communications between SRC1 and NOBL      DEGADIS1
c check4=t      steady state simulation                         DEGADISIN
c check5=t      operator sets sort parameters                   ESTRT3
c
      data CHECK1/.false./,CHECK2/.false./,AGAIN/.false./
      data CHECK3/.false./,CHECK4/.false./,CHECK5/.false./
C
      character*100 OPNRUP
      character OPNRUP1(100)
      equivalence (opnrup1(1),opnrup)
      character*4 in,ind,INP
      character*100 dummy
      DATA POUND/'//  '/,POUNDN/-1.E-20/
C
      DATA PTIME/igen*0.D0/
      DATA ET/igen*0.D0/,R1T/igen*0.D0/
      DATA PWC/igen*0.D0/,PTEMP/igen*0.D0/
      DATA PFRACV/igen*0.D0/,PENTH/igen*0.D0/
      DATA PRHO/igen*0.D0/
      data DEN/igen*0.,igen*0.,igen*0.,igen*0.,igen*0./

      DATA INP/'.INP'/,IN/'.IN '/,IND/'.IND'/
C
c
C... GET THE FILE NAME TO BE USED BY ALL OF THE ROUTINES.......................
C
      write(6,*) 'Beginning DEGBRIDGE...'

      READ(5,820) NCHAR,opnrup
      opnrup = opnrup(1:nchar) // ind(1:4)
C
C... First, get the desired information from RUN_NAME.IND
C
      open(unit=8,name=opnrup,type='old')
      read(8,*) oodist, cc, hwidth
c
c... If OODIST is zero, discontinue the run
c
      if(oodist .eq. 0.D0) then
            write(6,*) '   ...terminating the run since CCLOW is met.'
            dummy = opnrup(1:nchar)
            opnrup = opnrup(1:nchar) // '.com_xxx;1'
            open(unit=1,name=opnrup,type='unknown')
```

```
                opnrup = '$ copy '//opnrup(1:nchar)//'.out .lis'
                write(1,8020) (opnrup1(iii),iii=1,nchar+16)
                opnrup = '$ delete '//dummy(1:nchar)//'.com_xxx;1'
                write(1,8020) (opnrup1(iii),iii=1,nchar+19)
                opnrup = '$ stop'
                write(1,8020) (opnrup1(iii),iii=1,6)
                close(unit=1)
                opnrup = '@'// dummy(1:nchar) // '.com_xxx;1'//' '
                istat = lib$do_command(opnrup)
                write(6,*) ' ?DEGBRIDGE? EXIT failed to work.'
                stop
        endif
        close(unit=8)
C
C... Now, get the desired information from RUN_NAME.IN
C
        opnrup = opnrup(1:nchar) // in(1:4)
        open(unit=8,name=opnrup,type='old')

        read(8,8000) title(1)
        read(8,8000) title(2)
        read(8,8000) title(3)
        read(8,8000) title(4)
        read(8,*) u0, z0
        read(8,*) zr

        read(8,*) indvel, istab, rml
c
c... Based on INDVEL, set the Monin-Obukhov length as desired............
c
        if( indvel.ne.1 .or. indvel.ne.2) indvel = 1
        if( istab.le.0 .or. istab.gt.6) istab = 4
        if(indvel .eq. 1) then
                if(istab.eq.1) then
                        rml = -11.43D0 * zr**0.103D0
                else if(istab.eq.2) then
                        rml = -25.98D0 * zr**0.171D0
                else if(istab.eq.3) then
                        rml = -123.4D0 * zr**0.304D0
                else if(istab.eq.4) then
                        rml = 0.0D0
                else if(istab.eq.5) then
                        rml = 123.4D0 * zr**0.304D0
                else if(istab.eq.6) then
                        rml = 25.98D0 * zr**0.171D0
                endif
        endif

        read(8,*) tamb, pamb, relhum

        if( tamb.le.0. ) tamb = tamb+273.15D0
        if( pamb.gt.1.1 ) pamb = pamb/101325.D0
        if( relhum.lt.0. .or. relhum.gt.100. ) relhum = 50.
c
c... Calculate the absolute humidity HUMID
c
        vaporp = 6.0298D-3* exp(5407.D0*(1.D0/273.15D0 - 1.D0/tamb))
        sat = 0.622D0*vaporp/(pamb - vaporp)
        humid = relhum/100.D0 * sat

        read(8,*) tsurf
        if( tsurf.lt.250. ) tsurf = tamb

        read(8,8010) gasnam
        read(8,*) gasmw
        read(8,*) avtime
        read(8,*) TEMJET
        read(8,*) gasul, gasll, zll
        if( gasll.le.0. ) gasll = 0.01
        if( gasul.le.gasll ) gasul = dmax1( 1.1D0*gasll, 1.0D0)
c
c... Now that AVTIME has been set, the value of DELTAY can be fixed.  Also
c       set the values of BETAY
c
        goto(161,162,163,164,165,166) istab
 161    timeav = dmax1( avtime, 18.4D0)                      ! A
        deltay = 0.423D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        sigx_coeff = 0.02
        sigx_pow = 1.22
        sigx_min_dist = 130.
        goto 170
 162    timeav = dmax1( avtime, 18.4D0)                      ! B
        deltay = 0.313D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        sigx_coeff = 0.02
        sigx_pow = 1.22
        sigx_min_dist = 130.
        goto 170
 163    timeav = dmax1( avtime, 18.4D0)                      ! C
        deltay = 0.210D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        sigx_coeff = 0.02
        sigx_pow = 1.22
        sigx_min_dist = 130.
        goto 170
 164    timeav = dmax1( avtime, 18.3D0)                      ! D
        deltay = 0.136D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        sigx_coeff = 0.04
        sigx_pow = 1.14
        sigx_min_dist = 100.
        goto 170
 165    timeav = dmax1( avtime, 11.4D0)                         ! E
        deltay = 0.102D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        sigx_coeff = 0.17
        sigx_pow = 0.97
        sigx_min_dist = 50.
        goto 170
 166    timeav = dmax1( avtime, 4.6D0)                          ! F
        deltay = 0.0674D0*(timeav/600.D0)**0.2D0
        betay = 0.9D0
        sigx_coeff = 0.17
        sigx_pow = 0.97
        sigx_min_dist = 50.
c
 170    continue

c
c... Recover INDHT, CPK, and CPP.  If CPP is set to 0, then CPK contains........
c       the (constant) heat capacity.
c
        read(8,*) indht, CPK, cpp
        if(cpp .eq. 0.D0) then
                cpp = 1.D0
                CPK = CPK*gasmw - 3.33D4
        endif

c
c... recover NDEN and set ISOFL and DEN accordingly..............................
c
        read(8,*) nden
        if(nden .lt. -1) nden=-1

        if(nden .eq. -1) then
                isofl = 1
                rhoe = pamb*101325.D0*gasmw/8314.D0/TEMJET
                grhoe = rhoe
                rhoa = pamb*
     .              (1.D0+humid)/(0.002833D0 + 0.004553D0*humid)/tamb
                den(1,1) = 0.D0
                den(2,1) = 0.D0
                den(3,1) = rhoa
                den(4,1) = 0.D0
                den(5,1) = tamb
                den(1,2) = 1.D0
                den(2,2) = rhoe
                den(3,2) = rhoe
                den(4,2) = 0.D0
                den(5,2) = tamb
                den(1,3) = 2.D0
        else if(nden .eq. 0) then
                isofl = 0

                rhoe = pamb*101325.D0*gasmw/8314.D0/TEMJET
                grhoe = rhoe
                rhoa = pamb*
     *              (1.D0+humid)/(0.002833D0 + 0.004553D0*humid)/tamb

                wc = 1.D0
                wa = 0.D0
                enth = cpc(temjet)*(temjet - tamb)
                call setden(wc, wa, enth)

        else
                isofl = 1

                do iii = 1,nden
                read(8,*) den(1,iii), den(2,iii), den(3,iii)
                den(4,iii) = 0.D0
                den(5,iii) = tamb
                enddo
                den(1,nden+1) = 2.D0
                rhoe = den(3,nden)
                grhoe = rhoe
                rhoa = den(3,1)
        endif

        nden0 = nden
        if(nden .eq. -1) nden0 = 2


        read(8,*) erate
        read(8,*) elejet, DIAJET
        read(8,*) tend
        check4 = .true.
        if(tend .gt. 0.) check4 = .false.

        read(8,*) distmx

        close(unit=8)

c
c... It is time to prepare the input file for DEGADIS....................
c
        opnrup = opnrup(1:nchar) // inp(1:4)
        open(unit=8,name=opnrup,type='new')
c
c... TITLE
c
        DO 200 I=1,4
        WRITE(8,8000) TITLE(I)
```

```
  200   CONTINUE
C
c*** Atmospheric parameters:
c
        WRITE(8,1020) U0,Z0,ZR
C
c*** stability, averaging time for DELTAY, and derived parameters
c
        WRITE(8,1040) istab
        write(8,1020) oodist,avtime
        WRITE(8,1020) DELTAY,BETAy,rml
        WRITE(8,1020) sigx_coeff,sigx_pow,sigx_min_dist
c
c*** ambient pressure, temperatures, and humidity
c
        write(8,1025) tamb,pamb,humid
c
        ihtfl = indht
        htco  = 0.
        iwtfl = 0
        wtco  = 0.
c
        write(8,1060) isofl,tsurf
        write(8,1060) ihtfl,htco
        write(8,1060) iwtfl,wtco
C
c*** gas characteristics
c
        write(8,8010) gasnam
        WRITE(8,1020) gasmw, temjet,rhoe
        write(8,1020) cpk,cpp
        WRITE(8,1020) gasul,gasll,zll
c
c density curve if isothermal
c
        if(isofl .eq. 0) goto 460

        WRITE(8,1040) nden0
        DO 440 I=1,nden0
        WRITE(8,1025) DEN(1,I),DEN(2,I),DEN(3,I),Den(4,I),den(5,i)
  440   CONTINUE
C
c
 460    cclow = (gasll/2.D0) * pamb*101325.D0*gasmw/8314.D0/Tamb
        WRITE(8,1010) CcLOW
c
c*** source description FOR A DILUTED SOURCE ...
C
        call adiabat(0,wc,wa,yc,ya,cc,rho,wm,enth,temp)
c
        gmass0 = 0.            ! no initial cloud mass
```

```
      write(8,1020) gmass0
      np = 4
c
      if(check4) tend = 60230. ! [=] sec
c
      ess  = erate
      r1ss = hwidth
      slen = 2.D0*hwidth
      swid = pi*r1ss**2/slen/2.D0
      pwc(1) = wc
      ptemp(1) = temp
      pfracv(1) = 1.0
c
      PTIME(1) = 0.D0
      et(1)    = ess
      r1t(1)   = r1ss
      PWC(1)   = pwc(1)
      PTEMP(1) = ptemp(1)
      PFRACV(1)= 1.0D0

      PTIME(2) = tend
      et(2)    = ess
      r1t(2)   = r1ss
      PWC(2)   = pwc(1)
      PTEMP(2) = ptemp(1)
      PFRACV(2)= 1.0D0

      PTIME(3) = tend + 1.
      et(3)    = 0.
      r1t(3)   = 0.
      PWC(3)   = pwc(1)
      PTEMP(3) = ptemp(1)
      PFRACV(3)= 1.0D0

      PTIME(4) = tend + 2.
      et(4)    = 0.
      r1t(4)   = 0.
      PWC(4)   = pwc(1)
      PTEMP(4) = ptemp(1)
      PFRACV(4)= 1.0D0

      WRITE(8,1040) NP
      DO 800 I=1,NP
 800  WRITE(8,1030) PTIME(I),ET(I),R1T(I), PWC(I),PTEMP(I),PFRACV(I)
C
      write(8,*) check1,check2,again,check3,check4,check5
C
      istat = lib$date_time(tinp)
      write(8,1050) tinp
c
      if(check4) write(8,1020) ess,slen,swid          ! steady state
```

```
c
C
      CLOSE(UNIT=8)
C
c
  820 FORMAT(Q,A40)

 1010 format(1x,1pg14.7)
 1020 format(3(1x,1pg14.7))
 1025 format(5(1x,1pg14.7))
 1030 format(1x,5(1pg14.7,1x),1pg14.7)
 1040 format(1x,I5)
 1050 format(a24)
 1060 format(1x,i4,1x,1pg14.7)

 8000 format(a80)
 8010 format(a3)
 8020 format(80a1)
c

      CALL EXIT
      END
####
```

APPENDIX F

PARTIAL LISTING OF PROGRAM VARIABLES

Variable	Data Type	Symbol	Units	Comments
AGAIN	LOGICAL			Local communications in SSSUP
ALEPH	REAL			Collection of constants to calculate observer position and velocity
ALPHA	REAL	α	n/a	Power law velocity profile power
ALPHA1	REAL	$(1.0 + \alpha)$	n/a	
BETAY	REAL	β_y	n/a	Lateral similarity power
CCLOW	REAL		kg/m^3	Lowest mixture concentration of interest
CHECK1	LOGICAL			Unused logical flag
CHECK2	LOGICAL			When true, release type without a liquid source
CHECK3	LOGICAL			Local communications flag used in DEGADIS1
CHECK4	LOGICAL			When true, steady-state simulation
CHECK5	LOGICAL			When true, user sets time-sort parameters
DELTAY	REAL	δ_y	$m^{1-\beta}$	Lateral similarity coefficient
DEN(1,I)	REAL	y_c	mole fraction	Contaminant mole fraction
DEN(2,I)	REAL	c_c	kg/m^3	Contaminant concentration for the given mole fraction

Variable	Data Type	Symbol	Units	Comments
DEN(3,I)	REAL	ρ	kg/m^3	Mixture density for the given mole fraction
DEN(4,I)	REAL	h	J/kg	Mixture enthalpy for the given mole fraction
DEN(5,I)	REAL	T	K	Mixture temperature for the given mole fraction
EMAX	REAL		kg/s	Maximum of secondary source mass evolution rate
ESS	REAL	E	kg/s	Steady-state release rate
ET(I)	REAL	E(t)	kg/s	Source mass evolution rate as a function of time PTIME(I)
G	REAL	g	m/s^2	Acceleration due to gravity
GAMMAF	REAL	$\Gamma(1/(1+\alpha))$	n/a	
GAS_CPK	REAL	q_1	J/kmol K$^{p_1-1}$	Constant for contaminant heat capacity
GAS_CPP	REAL	P1	n/a	Power for contaminant heat capacity
GAS_LFL	REAL		mole fraction	Lower contaminant concentration level for estimating contours
GAS_MW	REAL	MW_C	kg/kmol	Contaminant molecular weight
GAS_NAME	CHARACTER*3			Name of contaminant
GAS_RHOE	REAL	ρ_0	kg/m^3	Saturated vapor density of contaminant at T_0
GAS_TEMP	REAL	T_0	K	Contaminant storage temperature
GAS_UFL	REAL		mole fraction	Upper contaminant concentration level for estimating contours

Variable	Data Type	Symbol	Units	Comments
GAS_ZSP	REAL		m	Height for estimating contours
GMASS0	REAL		kg	Initial mass of gas over the primary source
HTCO	REAL	h_0	J/m^2sK	Constant coefficient when IHTFL=-1
		V_H	m/s	LLNL heat transfer velocity when IHTFL=2
HUMID	REAL		kg water/ kg dry air	Ambient absolute humidity
IHTFL	INTEGER			Heat transfer flag: IHTFL=-1 constant coefficient IHTFL=0 no heat transfer IHTFL=1 DEGADIS correlation IHTFL=2 LLNL correlation
ISOFL	INTEGER			Isothermal release when ISOFL=1
ISTAB	INTEGER			Pasquill atmospheric stability indicator (ISTAB=1 for A, (ISTAB=2 for B, etc.)
IWTFL	INTEGER			Water transfer flag IWTFL=-1 constant coefficient IWTFL=0 no water transfer IWTFL=1 DEGADIS correlation
K	REAL	k	n/a	von Karman's constant, 0.35
LUNLOG	INTEGER			Fortran logical unit number which acts as a simulation log
MAXNOB	INTEGER			Maximum number of observers
ML	REAL	λ	m	Monin-Obukhov length
NOBS	INTEGER			Number of observers for the pseudosteady-state simulation

Variable	Data Type	Symbol	Units	Comments
NREC(I,1)	INTEGER			Number of records generated in PSSOUT for observer I
NREC(I,2)	INTEGER			Number of records generated in SSGOUT for observer I
PAMB	REAL	p	atm	Ambient pressure
POUND	CHARACTER*4			Character string to signal end of data ('// ')
POUNDN	REAL			Numerical value to signal end of data (-1.E-20)
QSTR(1,I)	REAL	t	s	Independent variable time for ordered pairs QSTR
QSTR(2,I)	REAL	Q_*	kg/m^2s	Atmospheric takeup rate as a function of time
RADG(1,I)	REAL	t	s	Independent variable time for ordered pairs RADG
RADG(2,I)	REAL	R	m	Secondary source radius as a function of time
RELHUMID	REAL		%	Ambient relative humidity
RHOA	REAL	ρ_a	kg/m^3	Ambient air density
RM	REAL	R_m	m	Radius at EMAX (when secondary source mass evolution rate is a maximum)
RMAX	REAL	R_{max}	m	Maximum secondary source radius
RT2	REAL	$\sqrt{2}.$	n/a	Constant
R1SS	REAL	R_p	m	Steady-state primary source radius

Variable	Data Type	Symbol	Units	Comments
R1T(I)	REAL	R_p	m	Primary source radius as a function of time PTIME(I)
SIGX_COEFF	REAL			Along-wind similarity coefficient
SIG_MIN_DIST	REAL		m	Minimum distance to apply x-direction dispersion correction
SIGX_POW	REAL		n/a	Along-wind similarity power
SLEN	REAL	L	m	Steady-state source length
SQPIO2	REAL	$\sqrt{\pi/2}$.	n/a	Constant
SQRTPI	REAL	$\sqrt{\pi}$	n/a	Constant
SRCDEN(1,I)	REAL	t	s	Independent variable time for ordered pairs SRCDEN
SRCDEN(2,I)	REAL	ρ	kg/m^3	Secondary source density as a function of time
SRCENTH(1,I)	REAL	t	s	Independent variable time for ordered pairs SRCENTH
SRCENTH(2,I)	REAL	h	J/kg	Secondary source enthalpy as a function of time
SRCWA(1,I)	REAL	t	s	Independent variable time for ordered pairs SRCWA
SRCWA(2,I)	REAL	w_a	mass fraction	Secondary source air mass fraction as a function of time
SRCWC(1,I)	REAL	t	s	Independent variable time for ordered pairs SRCWC

Variable	Data Type	Symbol	Units	Comments
SRCWC(2,I)	REAL	w_c	mass fraction	Secondary source contaminant mass fraction as a function of time
SWID	REAL		m	Steady-state source half-width
SZM	REAL	S_{z0m}	m	Value of S_{z0} at EMAX (when secondary source mass evolution rate is a maximum)
TAMB	REAL	T	K	Ambient temperature
TEND	REAL		s	Termination time of secondary source
TINP	CHARACTER*24			Time DEGADISIN was executed
TITLE(1:4)	CHARACTER*80			Text title block 4 lines of 80 spaces
TO(I)	REAL		s	Time of release for observer I
TSURF	REAL	T_s	K	Surface temperature
USTAR	REAL	u_*	m/s	Friction velocity
UO	REAL	u_0	m/s	Ambient velocity at height z_0
WTCO	REAL	F	kg/m^2s	Mass transfer coefficient when IWTFL=-1
XV(I)	REAL	x_v	m	Virtual source position for estimation of S_y in SSG
ZO	REAL	z_0	m	Height for velocity u_0
ZR	REAL	z_R	m	Roughness length

APPENDIX G
DEGADIS DIAGNOSTIC MESSAGES

To assist the user in determining the source of any problems, a diagnostic procedure has been included in DEGADIS. The subroutine TRAP is meant to cause an orderly termination of the program for many detected errors. It performs two basic functions: TRAP displays an error code and a single line diagnostic message giving the reason for premature termination, and TRAP forces an output of the COMMON area data sets to the file TRAP.DBG.

The first three lines sent to the execution log (default-TERMINAL) include the TRAP introductory lines and the error code number:

```
The best laid plans of mice and men . . .
You have entered a TRAP--THE LAND OF NO RETURN
CODE:  NN
```

where NN represents the code of the error message which follows in the log. The error message begins with the name of the calling routine.

The following is a list of the error codes, error messages, and suggested actions for each problem.

Code: 1
DEGADIS1? Source integration has returned IHLF=NN

Action: This error occurs during integration of the equations which describe the gas source. IHLF is an error code returned by the integration package RKGST.

When IHLF=11, more than 10 bisections of the initial increment of the independent variable were necessary to make an integration step within the specified error. Reduce the initial step size of the independent variable (STPIN in the ER1 file). If this does not work, it will be necessary to either increase the error criteria for all of the dependent variables being integrated (ERBND in the ER1 file) or increase the error criteria for the variable violating the criteria by decreasing the error weight for that variable (one of the following: WTRG, WTTM, WTYA, WTYC, WTEB, WTMB, or WTUH in the ER1 file).

When IHLF=12, the initial increment of the independent variable (STPIN) is 0. Correct the ER1 file and execute the program again.

When IHLF=13, the initial increment of the independent variable (STPIN) is not the same sign as the difference between the upper bound of the interval and the lower bound of the interval. STPIN must be positive. Correct the ER1 file and execute the program again.

Code: 2
Reserved

Code: 3
SZF? Local integration failed; IHLF=NN

Action: This error occurs during estimation of SZ over the source when no gas is present. IHLF is an error code returned by the integration package RKGST.

When IHLF=11, more than 10 bisections of the initial increment of the independent variable were necessary to make an integration step within the specified error. Reduce the initial step size of the independent variable (SZSTP0 in the ER1 file). If this does not work, increase the error criteria for all of the dependent variables being integrated (SZERR in the ER1 file).

When IHLF=12, the initial increment of the independent variable (SZSTP0) is 0. Correct the ER1 file and execute the program again.

When IHLF=13, the initial increment of the independent variable (SZSTP0) is not the same sign as the difference between the upper bound of the interval and the lower bound of the interval. SZSTP0 must be positive. Correct the ER1 file and execute the program again.

Code: 4
SURFACE? Negative QRTE for positive DELTA_T.

Action: This is a diagnostic message indicating an error in estimation of the heat capacity. Check the input to the model and execute the program again.

Code: 5
CRFG? More points for GEN3 were needed.

Action: The COMMON area /GEN3/ stores representative values of the calculated source parameters. If this message occurs, relax the CRFG error criteria (CRFGER) in the ER1 file. If this is a common problem, the length of the /GEN3/ vectors can be increased by changing the value of MAXL in DEGADIS1.DEC and reinstalling DEGADIS.

Code: 6
TUPF? Observer calculations--TUPF failed

Action: The trial-and-error search associated with finding the upwind edge of the gas source for an observer failed. Often this problem can be avoided by adding one or two additional observers to the present number of observers (which changes the conditions for the trial and error). Another possibility is to increase the error criteria for this function (ERTUPF) in the ER2 file.

Code: 7
TDNF? Observer calculations--TDNF failed

Action: The trial-and-error search associated with finding the downwind edge of the gas source for an observer failed. Often this problem can be solved by adding one or two additional observers to the present number of observers (which changes the conditions for the trial and error). Another possibility is to increase the error criteria for this function (ERTDNF) in the ER2 file.

Code: 8
SSSUP? Observer Integration failed, IHLF=NN

Action: This error occurs during integration of the five differential equations which average the source for each observer. IHLF is an error code returned by the integration package RKGST.

When IHLF=11, more than 10 bisections of the initial increment of the independent variable were necessary to make an integration step within the specified error. Reduce the initial step size of the independent variable (STPO in the ER2 file). If this does not work, it will be necessary to either increase the error criteria for all of the dependent variables being integrated (ERRO in the ER2 file) or increase the error criteria for the variables violating the criteria by decreasing the error weight for that variable (one of the following: WTAIO, WTQOO, or WTSZO in the ER2 file).

When IHLF=12, the initial increment of the independent variable (STPO) is 0. Correct the ER2 file and execute the program again.

When IHLF=13, the initial increment of the independent variable (STPO) is not the same sign as the difference between the upper bound of the interval and the lower bound of the interval. STPO must be positive. Correct the ER2 file and execute the program again.

Code: 9
SSSUP/SDEGADIS2? Pseudosteady Integration failed, IHLF=NN

Action: This error occurs during integration of the four differential equations describing the portion of the downwind calculation when b > 0. The routine calling TRAP is SSSUP if a transient simulation is being executed; if a steady-state simulation is being executed, the calling routine is SDEGADIS2. IHLF is an error code returned by the integration package RKGST.

When IHLF=11, more than 10 bisections of the initial increment of the independent variable were necessary to make an integration step within the specified error. Reduce the initial step size of the independent variable (STPP in the ER2 file). If this does not work, it will be necessary to either increase the error criteria for all of the dependent variables being integrated (ERRP in the ER2 file) or increase the error criteria for the variable violating the criteria by decreasing the error weight for that variable (one of the following: WTSZP, WTSYP, WTBEP, or WTDH in the ER2 file).

When IHLF=12, the initial increment of the independent variable (STPP) is 0. Correct the ER2 file and execute the program again.

When IHLF=13, the initial increment of the independent variable (STPP) is not the same sign as the difference between the upper bound of the interval and the lower bound of the interval. STPP must be positive. Correct the ER2 file and execute the program again.

Code: 10
SSSUP/SDEGADIS2? Gaussian Integration fail, IHLF=nn

Action: This error occurs during integration of the differential equations describing the portion of the downwind calculation when b=0. The routine calling TRAP is SSSUP if a transient simulation is being executed; if a steady-state simulation is being executed, the calling routine is SDEGADIS2. IHLF is an error code returned by the integration package RKGST.

When IHLF=11, more than 10 bisections of the initial increment of the independent variable were necessary to make an integration step within the specified error. Reduce the initial step size of the independent variable. (STPG in the ER2 file). If this does not work, it will be necessary to either increase the error criteria for all of the dependent variables being integrated (ERRG in the ER2 file) or increase the error criteria for the variable violating the criteria by decreasing the error weight for that variable (either WTRUH or WTDHG in the ER2 file).

When IHLF=12, the initial increment of the independent variable (STPG) is 0. Correct the ER2 file and execute the program again.

When IHLF=13, the initial increment of the dependent variable (STPG) is not the same sign as the difference between the upper bound of the interval and the lower bound of the interval. STPG must be positive. Correct the ER2 file, and execute the program again.

Code: 11
SSSUP/SDEGADIS2 Total No. of Records exceeds 120,000

Action: This is an arbitrary stopping point for the process in order to keep a runaway simulation from filling up disk space. Relax the output specification (ODLP, ODLLP, ODLG, or, DLLG) in the ER2 file in order to generate less output if the input parameters are valid.

Code: 12
Reserved

Code: 13
Reserved

Code: 14
Reserved

Code: 15
Reserved

Code: 16
PSSOUT/PSSOUTSS? PSS started with $B < 0$.

Action: This condition is checked at the beginning of the downwind calculation in order to confirm proper handling of the movement to the Gaussian phase of the downwind calculation. Check the initial conditions and execute the program again.

Code: 17
TPROP/ADDHEAT? Enthalpy out of bounds

Action: Diagnostic message indicating that an enthalpy lower than the adiabatic mixing enthalpy was passed to ADDHEAT. Check the input conditions and execute the program again.

Code: 18
ALPH? ALPHA integration failed, IHLF=NN

Action: The integration which determines the integral least squares fit for ALPHA has failed. Note that small values of the Monin-Obukhov length (ML < O(1m)) in combination with stable atmospheric conditions may cause this failure. IHLF is an error code returned by the integration package RKGST.

When IHLF=11, more than 10 bisections of the initial increment of the independent variable were necessary to make an integration step within the specified error. Reduce the absolute value of the initial step size of the independent variable (STPINZ in the ER1 file). If this does not work, it will be necessary to increase the error criteria (ERBNDZ in the ER1 file).

When IHLF=12, the initial increment of the independent variable (STPINZ) is 0. Correct the ER1 file and execute the program again.

When IHLF=13, the initial increment of the independent variable (STPINZ) is not the same sign as the difference between the upper bound of the interval and the lower bound of the interval. STPINZ must be negative. Correct the ER1 file and execute the program again. This error will also occur if the surface roughness ZR is greater than the reference height Z0.

Code: 19
ALPH? RTMI has failed to locate ALPHA IERR:NN

Action: The search procedure which determines ALPHA has failed. This error may be the result of an unusual velocity specification such as small values of the Monin-Obukhov length (ML < O(1.m)) or small reference heights (Z0 < O(10. * ML)). IERR is an error code returned by the routine RTMI.

When IERR=1, the search for ALPHA failed after a specified number of iterations. Increase the error bound used by RTMI (EPS in the ER1 file).

When IERR=2, the basic assumption that the function which governs the search for ALPHA changes sign over the specified interval is false. Increase the search interval by decreasing the lower bound of ALPHA (XLI in the ER1 file) and increasing the upper bound (XRI in the ER1 file).

Code: 20
ESTRT? Premature EOF in RUN_NAME.ER1 or RUN_NAME.ER2

Action: The portion of the program which reads ER1 and ER2 files encountered an end-of-file mark before all of the information had been read. Confirm these files and execute the program again. If necessary, copy and edit the appropriate EXAMPLE file and execute the program again.

Code: 21
ESTRT1/ESTRT2/ESTRT2SS/ESTRT3? DECODE failed.

Action: The portion of the program which reads the ER1, ER2, or the ER3 file failed to understand a numerical entry. The numbers must appear in columns 11-20 of the line with no alphabetic characters in the field. (Note that exponential notation is not allowed.) This restriction does not apply to comment lines which have an exclamation point (!) in the first column.

Code: 22
ESTRT1? The parameter file RUN_NAME.ER1 was not found.

Action: The ER1 file was not found for the current simulation (RUN_NAME). Copy the file EXAMPLE.ER1 to RUN_NAME.ER1 and edit it as necessary. Execute the program again.

Code: 23
SORTS1? Fewer than 3 points sorted for any time.

Action: Only one or two simulation points were applicable for the sort times specified. There are three possible causes for this condition:

(1) if this message appears when the sort times are defaulted (CHECK5 is set to 0. in the ER3 file), the number of observers will probably have to be increased to give a good resolution of the downwind concentration field. (The number of observers is NOBS in the ER2 file with a maximum (MAXNOB) given in DEGADIS2.DEC.). As a rule of thumb, one gets good resolution of the downwind concentration field if the ratio: (secondary source duration / number of observers) is less than about 10 seconds (or 20 at most).

(2) The sort times specified in the ER3 file were before the simulation had developed significantly. This is only applicable when the user is specifying the sort times (i.e. when CHECK5 is set to 1. in the ER3 file). Increase the time of the first sort (ERT1), and rerun the program.

(3) The sort times specified in the ER3 file were after the gas was below the lowest concentration of interest. This is only applicable when the user is specifying the sort times (i.e. when CHECK5 is set to 1. in the ER3 file). Increase the time of the first sort (ERT1), and rerun the program. If additional results are desired for later times, restart the simulation and specify a lower concentration of interest in the input step (lower CCLOW in DEGADISIN).

Code: 24
TPROP? Trial and error loop compromised.

Action: TPROP estimates the temperature of a mixture based upon the composition and enthalpy of the mixture. Ensure the properties for the diffusing species are entered correctly and execute the simulation again.

Code: 25
TPROP? Isothermal density loop compromised.

Action: This error should never occur, but if it does, rebuild the model from the original files and run the simulation over.

Code: 26
TPROP? Invalid entry flag in ADIABAT.

Action: This is a programming diagnostic and should never occur. If it does, rebuild the model from the original files.

Code: 27
Reserved

Code: 28
TPROP? IGEN request too large in SETDEN.

Action: The subroutine SETDEN (in TPROP) performs a series of adiabatic mixing calculations with a specified gas mixture and ambient air and places the result in the array DEN(5,IGEN). This error indicates more points are needed in DEN than were originally requested. Increase the allocation for DEN by changing the value of IGEN in DEGADISIN.DEC and reinstalling DEGADIS.

Code: 29
PHIF? Flag IPHIFL is out of bounds.

Action: Proper values of IPHIFL are integers between 1 and 5 inclusive. Although values of IPHIFL are entered in the ER1 file as real numbers, they should be in this range. Check the ER1 file and execute the program again.

Code: 30
SSSUP/SDEADIS2? Concentration greater than RHOE.

Action: If the concentration of the contaminant becomes greater than the pure component density for an isothermal simulation, this error will occur. However, this situation should never occur. Check the input conditions and execute the program again.

Code: 31
SSSUP? Concentration greater than RHOE.

Action: If the concentration of the contaminant becomes greater than the pure component density for an isothermal simulation, this error will occur. However, this situation should never occur. Check the input conditions and execute the program again.

Code: 32
PSS? Sz convergence failure.

Action: This is a programming diagnostic and should never occur. If it does, check the input conditions and execute the program again.

Code: 33
SSG? Sz convergence failure.

Action: This is a programming diagnostic and should never occur. If it does, check the input conditions and execute the program again.

(This page is intentionally left blank.)